

ECP-2008-DILI-518001

BHL-Europe

**Key components documented for output of D3.5
e.g. BHL-Europe Portal, OCR demonstrators,
distributed storage model, etc.**

Deliverable number	<i>D3.7</i>
Dissemination level	<i>Public</i>
Delivery date	<i>14 May 2011</i>
Status	<i>Final</i>
Author(s)	<i>TMB</i>



eContentplus

This project is funded under the *eContentplus* programme¹,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

¹ OJ L 79, 24.3.2005, p. 1.

TABLE OF CONTENTS

1	DOCUMENT HISTORY	4
1.1	CONTRIBUTORS	4
1.2	REVISION HISTORY	4
1.3	REVIEWERS AND APPROVALS	4
1.4	DISTRIBUTION.....	5
2	PURPOSE AND DOCUMENT STRUCTURE.....	6
2.1	OVERVIEW – THE BHL-EUROPE KEY COMPONENTS	6
3	INTERDEPENDENCIES BETWEEN KEY COMPONENTS	9
3.1	OAIS COMPONENT DEPLOYMENT	9
3.1.1	<i>Execution Environments.....</i>	9
3.1.2	<i>Filetype.....</i>	10
3.1.3	<i>OAIS Components</i>	10
4	KEY COMPONENTS.....	11
4.1	PRE PRE-INGEST METADATA MAPPING TO OLEF	11
4.2	PRE-INGEST	13
4.2.1	<i>Workflow</i>	14
4.2.1.1	Submission of Information Packages	14
4.2.1.2	Processing of submitted Information Packages	15
4.2.1.3	Preparation of Archival Information Package	16
4.2.2	<i>Pre-Ingest in context of the OAIS function model.....</i>	16
4.2.3	<i>Data Harmonization.....</i>	17
4.2.3.1	Pre-Ingest Tool.....	18
4.2.4	<i>Metadata Enrichment.....</i>	21
4.3	INGEST.....	23
4.4	ARCHIVAL STORAGE	25
4.4.1	<i>The Fedora Digital Object.....</i>	27
4.4.2	<i>Islandora Content Model</i>	27
4.4.3	<i>Deployment</i>	27
4.5	DATA MANAGEMENT.....	28
4.5.1	DEPLOYMENT	30
4.6	ACCESS.....	30
4.7	PORTAL.....	32



4.7.1	<i>Simple Search</i>	33
4.7.2	<i>Advanced Search</i>	34
4.7.3	<i>Search Results View</i>	37
4.7.4	<i>Metadata View</i>	38
4.7.5	<i>User Profile Personalization</i>	39
4.7.6	<i>Multi-lingual UI of the BHL-Europe Portal</i>	43
4.7.7	<i>Content View</i>	47
4.7.8	<i>Browse Index</i>	48
4.7.9	<i>Drupal 7</i>	48
4.7.9.1	Modules.....	49
4.7.9.2	Nodes.....	49
4.7.9.3	Fields.....	49
4.7.9.4	Drupal API.....	49
4.7.9.5	Solr integration into Drupal 7.....	49
4.7.9.6	XML Stylesheets.....	50
4.8	GUID MINT/RESOLVER.....	50
5	THE INGESTION PROCESS FROM BHL-EUROPE TO EUROPEANA	51
5.1	THE EUROPEANA FIELDS.....	52
5.2	EUROPEANA CONTENT CHECKING.....	56
I	ACRONYMS	59
II	FIGURES	61
III	APPENDICES	64
III.I	PRE-INGEST FILE SUBMISSION GUIDELINES.....	64
III.II	TECHNOTE: WEBDAV.....	74
III.III	TECHNOTE: BHL-EUROPE GUID.....	82
III.IV	TECHNOTE: DATAFLOW.....	88
III.V	TECHNOTE: LOCKSS.....	105
III.VI	TECHNOTE: FEDORA-COMMONS.....	114
III.VII	TECHNOTE: ISLANDORA.....	155

1 Document History

This chapter describes the document's creation events and contributors.

1.1 Contributors

This document is based on the meetings in Egypt, Berlin and Graz and ongoing progress and outputs of the technical team with the following members contributing to this document.

Person	Partner
Walter Koch, Bernd Sproger, Gerda Koch, Johannes Edelsbrunner	AIT
Lee Namba	ATOS
Chris Sleep, Lola Obajuluwa, Adrian Smales	NHM
Wolfgang Koller	NHMW
Henning Scholz	MfN

1.2 Revision History

Revision Date	Author	Version	Change Reference & Summary
25 April 2011	TMB	0.1	1 st Component Draft
28 April 2011	TMB	0.2	Revised components assembled
29 April 2011	TMB	0.3	Components revised and updated
2 May 2011	TMB	0.4	Draft taking internal review into account
14 May 2011	TMB	Final	

1.3 Reviewers and Approvals

This document requires the following reviews and approvals. Signed approved forms are kept in the project file.

Name	Position	Date	Version
Adrian Smales / Graham Higley	Work Package Leader	14 May 2011	Final
BHL-Europe Tech Group	TMB		0.4
Henning Scholz	Project Co-ordinator		0.4

1.4 Distribution

This document has been distributed to:

Group	Date of issue	Version
BHL-Europe Tech Group	28 April 2011	0.2
BHL-Europe Tech Group	29 April 2011	0.3
BHL-Europe Tech Group	2 May 2011	0.4
BHL-Europe consortium	4 May 2011	0.4
BHL-Europe Tech Group	16 May 2011	Final
Project Co-Ordinator	16 May 2011	Final

2 Purpose and document structure

This aim of this document is to describe the technical architecture, workflows and interdependencies for the Key Components identified in deliverable D3.5 (Technical architecture status and progress report with particular focus on the development of the German prototype) with specific implementation details for the components deployed to date.

Chapter 4 of the document structure reflects the structure of D3.4/D3.5 covering the component implementation and process/data flows. It is followed by a chapter describing the current ingestion process from BHL-Europe to Europeana, achieved for the data sets ingested during April 2011.

Overall this document will cover the current technical architecture status in terms of where our technology stands so far in the Implementation Phase.

The final development will be delivered in D3.9

2.1 Overview – The BHL-Europe Key Components

BHL-Europe Technical Architecture

- Pre-Ingest
- Ingest
- Archival Storage
- Data Management
- Access
- Portal

The BHL-Europe data workflow is based on systems to separate roles and responsibilities and to effectively manage the data. The outcome of every individual system is feeding the Pre-Ingest module of the BHL-Europe system. The Pre-Ingest module thus collects all the data produced in these systems and prepares the SIP to be ingested into BHL-Europe. Among these systems are the scanning system and the mapping system (called schema mapping in the architecture). The scanning system on the content provider side will provide the page images. The mapping system will provide the metadata mapped to the OLEF (Open Literature Exchange Format). No metadata other than data in the OLEF schema is allowed to enter the Pre-Ingest module. It is of special importance that the archived information can reconstruct exactly the same output OLEF as the one which was submitted to the Pre-Ingest module.

BHL-Europe is offering the metadata mapping to the OLEF schema as a service for all content providers (for details on this process see section 4.1 below). This does not disengage content providers to be responsible for the quality of their metadata, as we don't have the resources to fix all quality issues. To start the process, every content provider is uploading metadata and scans via FTPS to the NHM servers in London.

Content Providers first upload test content to the NHM server including the scanned images and the corresponding metadata according to the file submission guidelines. The test content

will then be checked on the one hand by AIT and on the other hand by NHMW. AIT verifies if the file submission guidelines have been followed and NHMW will start with the data mapping to the OLEF schema creating a configuration for every content provider to accommodate for the characteristics of the content provider data. The mapping configuration is defined for every content provider individually once and will remain the same as long as the content provider does not change the structure of the provided metadata. However, standard mappings (for e.g. MARC21, DC) are available and only partners with non-standard data need more work to map the data properly. According to our preliminary analysis, this is only true for a low number of partners. The mapping process is accompanied by a QA process to negotiate and verify the result of the mappings in collaboration with the content providers. After quality is approved, the content provider can upload the bulk of the material to be ingested into the BHL-Europe system. Once the upload is finished, the content provider log into the Pre-Ingest module to trigger the conversion of all metadata into the OLEF schema based on the previously defined configuration. NHMW provides the command to convert the metadata for each content provider which is then executed by the Pre-Ingest module. The content provider then selects the folder(s) for procession via the Pre-Ingest interface. The data handling in the Pre-Ingest module will be managed by AIT.

In case content providers have the expertise and resources to work on the mapping configuration themselves, we can offer this options and provide all the necessary tools for the content providers. The Schema Mapping Tool is open source and documentation will be available before the end of the project. This also is a sustainable solution to distribute the responsibility and offer future content providers tools to map their data to our data schema after the end of the project.

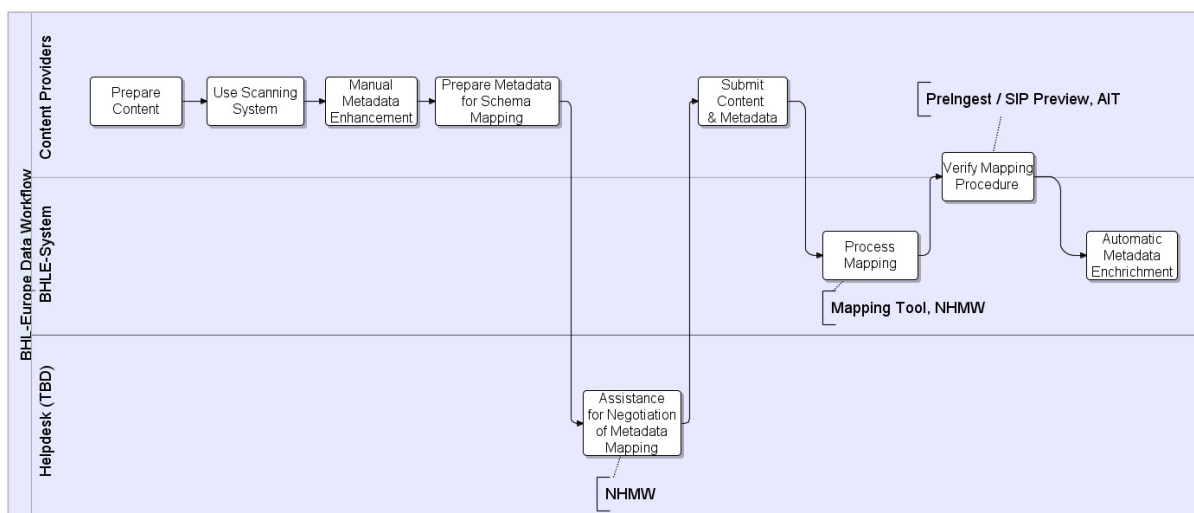


Figure 2-1: SIP Preparation Workflow

The Schema Mapping Tool is also able to deal with enriched metadata of our content providers and include this information in the OLEF schema. Manual data enrichment ideally happens at time of scanning and during quality control on the content provider side. This may include taxonomic information in high quality. It needs to be investigated, if workflow management tools like Goobi are able to support the data enrichment procedure.

If one content provider, for example, is able to provide good quality taxonomic metadata for the page images or scans, BHL-Europe is able to use these data to facilitate search and retrieval of these items. In any case an automatic enrichment of the metadata will be done



during Pre-Ingest using live Web-Services. Pre-processing of the data is required to facilitate the automatic data enrichment. OCR is necessary, for example, to apply the TaxonFinder and identify taxon names in the documents provided to BHL-Europe. Within the Pre-Ingest module, METS files are generated as a transport container (encapsulating the OLEF metadata) for Fedora. Also the mapping to the Europeana schema (ESE, EDM) is part of the Pre-Ingest and is done by AIT. For more details on the Pre-Ingest component, please see section 4.2 below.

3 Interdependencies between Key Components

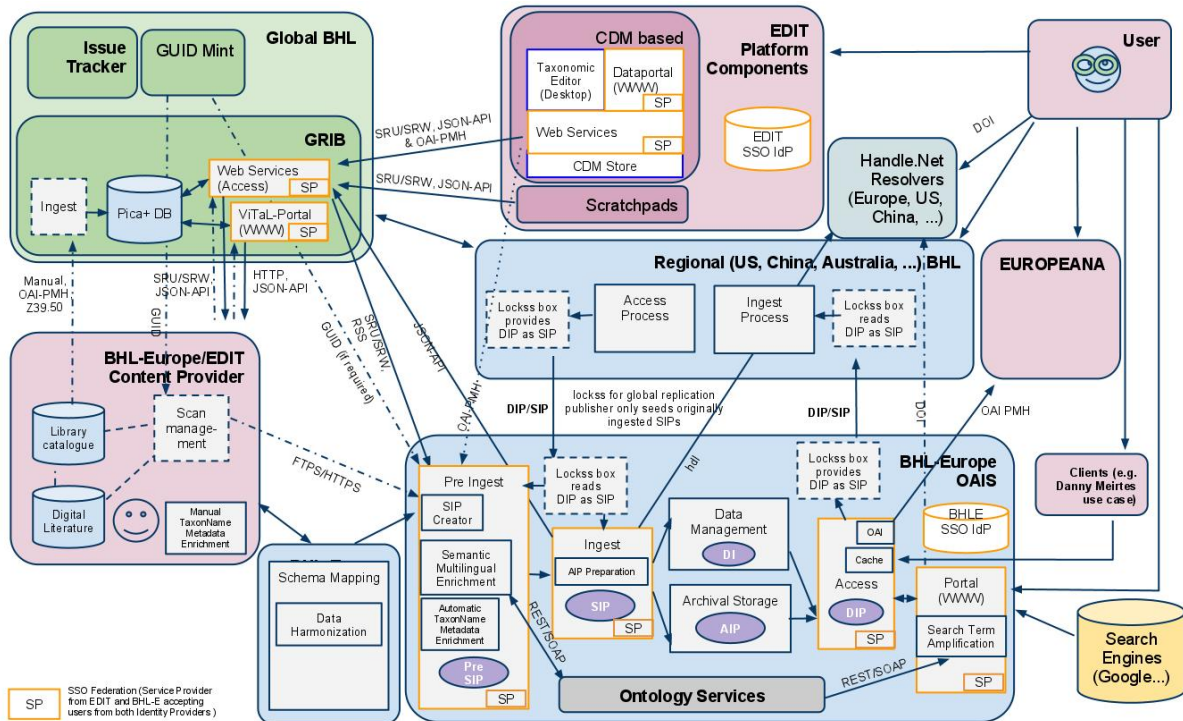


Figure 3-1: BHL-Europe Architecture Diagram

3.1 OAIS Component Deployment

BHL-Europe uses virtualization technology from VMWare whereby larger physical servers are used in place of many small physical servers to increase the utilization of costly hardware resources such as CPU. New virtual machines can be provisioned as needed for scalability or redundancy needs. In order to reduce configuration and maintenance efforts all software specific to an OAIS component is installed on the shared network working storage. This storage is organized by execution environments, filetype, and OAIS components.

3.1.1 Execution Environments

In order to maintain a stable system capable of evolving in the future several execution environments are needed. At a minimum development and production environments are required to have a stable system in production and a development system for bug fixes, feature development and testing.

Execution Environment	Path to Environment
Development	/mnt/nfs-demeter/dev
Staging	/mnt/nfs-demeter/stage
Production	/mnt/nfs-demeter/prod

3.1.2 Filetype

On order to optimize the performance and maintainability of the BHL-Europe system the files are separated into three types based on typical best practices of Linux systems:

- Software engines
- Data files of software engines
- Log files of software engines

Using the production environment as an example this gives the following:

Type of File	Path to Filetype
Software engine	/mnt/nfs-demeter/prod/opt
Data	/mnt/nfs-demeter/prod/data
Logs	/mnt/nfs-demeter/prod/logs

3.1.3 OAIS Components

To keep the BHL-Europe system modular the OAIS components are separated as well. Using the production environment and the software engines as examples this gives the following:

OAIS Component	Path to OAIS Software Components
Pre-ingest	/mnt/nfs-demeter/prod/opt/pre-ingest
Ingest	/mnt/nfs-demeter/prod/opt/ingest
Archival Storage	/mnt/nfs-demeter/prod/opt/archival-storage
...	...

4 Key Components

4.1 Metadata mapping to OLEF

As content providers use different metadata for their exports it was necessary to introduce an additional processing step for harmonizing the metadata. In order to provide a common input format to all further processing steps it was decided to use OLEF (Open Literature Exchange Format). The format will be explained more in detail later on.

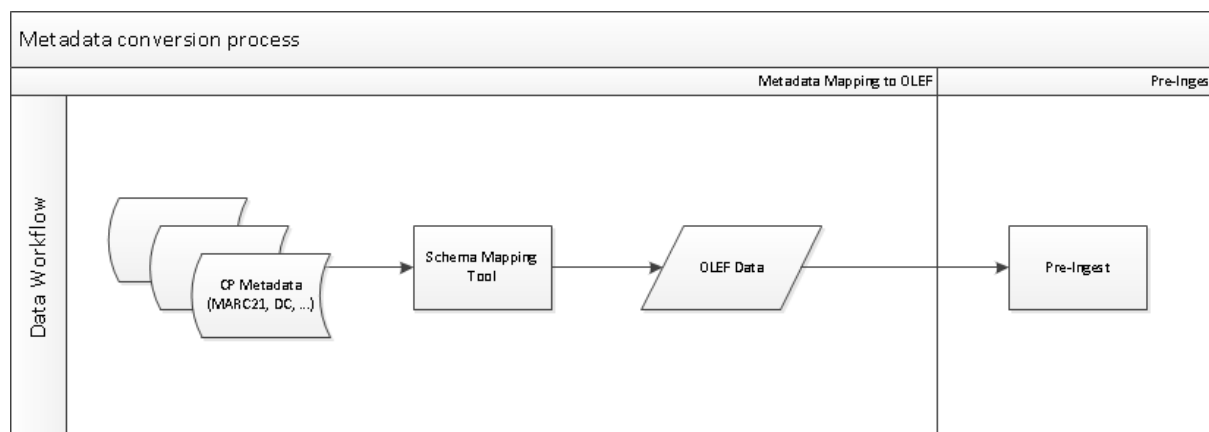


Figure 4-1: Metadata Mapping to OLEF

As outlined in Figure 4-1 all incoming metadata will be pre-processed using the SMT (Schema Mapping Tool). The SMT has several built in conversion templates for processing standard input formats (like MARC21, DC, MODS, etc.). Some content providers may provide their metadata using a special format. Therefore the SMT is also able to process custom mappings which can be pre-defined using a graphical user interface.

The metadata mapping to OLEF is a preparation phase for the Pre-Ingest module which however is directly invoked by the Pre-Ingest itself. The conversion of the metadata can be actively triggered by the Content Provider itself when accessing the Pre-Ingest interface. Therefore the metadata mapping to OLEF is more a conceptual step than a physically separated process.

As a first step we defined our requirements involving all of our content providers. In order to allow communication between all of our partners we started a Google group (BHLE-Metadata) which primary task it was to create a simple list of requirements for our metadata including a comparison to existing standards. During the gathering of the requirements for the metadata it was discovered that no existing standard can fulfill all desired requirements on its own. Therefore we decided to create an own schema which encapsulates several domain specific standards into a single exchange and storage format which satisfies all of our requirements.

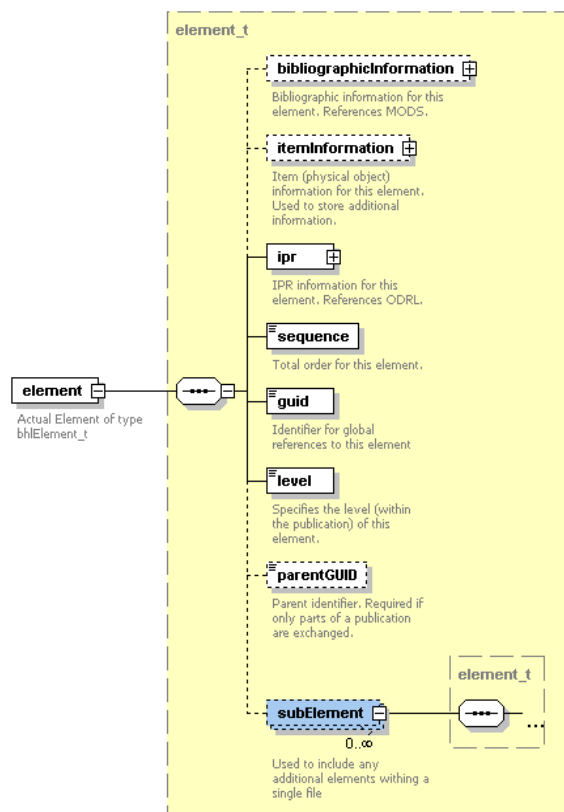


Figure 4-2: Schematic outline of OLEF structure

As shown in Figure 4-2 the OLEF schema uses a simple yet effective approach for storing literature related metadata in a single file. The main problem for all existing standards is the lack of ability to store metadata at any given level within the hierarchy of a publication. The OLEF schema tackles that limitation by introducing a recursive element (called “element” and “subElement” in Figure 4-2) which is able to reference itself within the same structure. Using that technique it is now possible to store metadata at any given level of information (monographs, serials, articles, pages, etc). Even only parts of a publication can be exchanged as the OLEF schema offers fields for storing and referencing GUIDs which are used for each item within the BHL-Europe architecture.

As mentioned above the OLEF schema incorporates several other standards and creates a relation between them. The standards used are:

- MODS² – bibliographic metadata (author, title, publication date, etc.)
- MIX³ – Image (scans) metadata (resolution, size, acquisition device, etc.)
- ODRL⁴ – IPR information
- DWC⁵ – Taxon Name information

² Metadata Object Description Schema: <http://www.loc.gov/standards/mods/>

³ Metadata for Images in XML Schema: <http://www.loc.gov/standards/mix/>

⁴ Open Digital Rights Language: <http://odrl.net/>

The combination of those standards has proven to satisfy all of our requirements. As of OLEF version 0.3, a stable namespace for the schema has been established using the BHL-Europe Web site. The current version of the schema can be found at <http://www.bhl-europe.eu/bhl-schema/v0.3/> (which is equal to the namespace).

The conversion process is already established at the servers provided by NHM London and is actively tested using the first uploads by our content providers. The integration into the Pre-Ingest module workflow is outlined in the next chapter.

4.2 Pre-Ingest

The Pre-Ingest component is a set of processing steps and processing systems which are orchestrated to facilitate data submission, data harmonization, data enrichment. This also involves communication and feedback loops with data providers and developers. Further documentation of this key component is referring to a workflow diagram which has been modeled based upon analysis of the required processing steps and capabilities of the used processing systems. “This component is the interface to the archives and acts as an adapter for the Ingest module. As external partners store meta data in various formats, the native formats need to be converted, harmonized, enriched and prepared for ingestion. This step is needed for the ingestion, multilingual search, data harmonization, indexation and search requirement.” [D3.4, p.18]

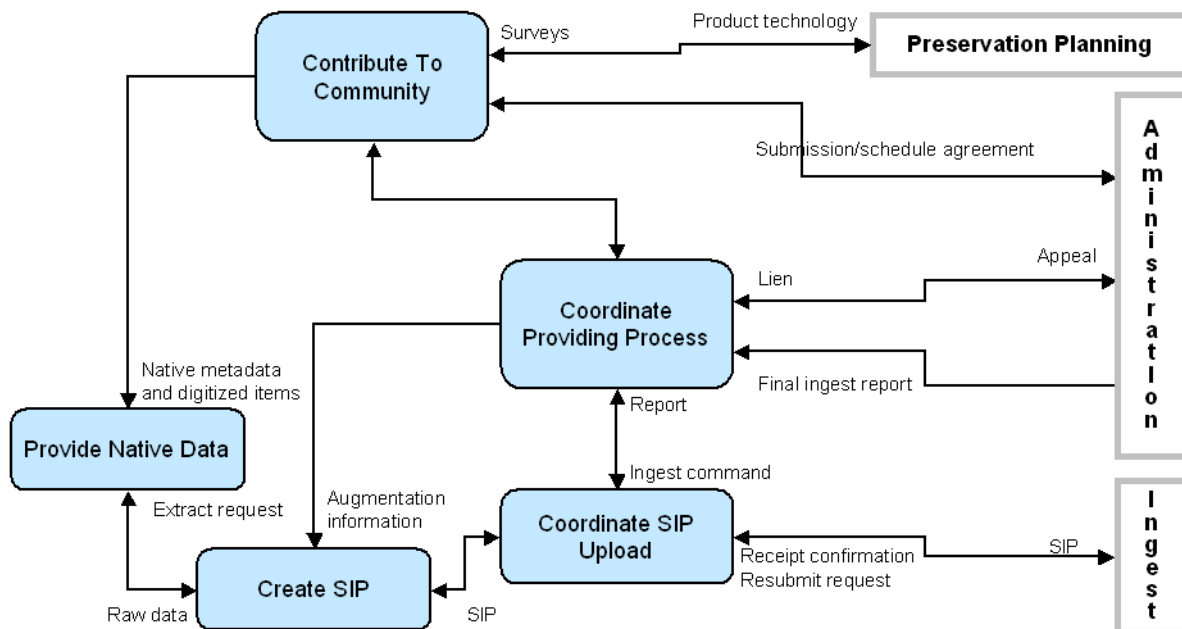


Figure 4-3: Pre-Ingest process as documented in Deliverable 3.4

The Pre-Ingest process as documented in D3.4 has been more precisely formulated by creating a Pre-Ingest workflow model.

⁵ Darwin Core: <http://rs.tdwg.org/dwc/>

4.2.1 Workflow

The Pre-Ingest workflow⁶ describes the necessary interaction steps to create an Archival Information Package (AIP) from a Submission Information Package (SIP). We have carefully investigated other best-practice approaches on how to setup a Pre-Ingest workflow. Pre-Ingest development is based on the works of Archivemata and the California Digital Library (CDL). CDL develops micro-services for several functionalities needed during Pre-Ingest. Also Archivemata developed an OAIS-based archival process during which different tools and micro-services are used. We compared both approaches and isolated important micro-services which are relevant for Pre-Ingest (see Figure 4-4). The resulting workflow model can be seen as a best-practice based approach of how a Pre-Ingest process generally could work.

Micro-Services (Merritt vs. Archivemata)

Micro-Services
for Pre-Ingest

CDL/UC3-MERRITT		ARCHIVEMATICA
Value ----- Service ----- Curation ----- Preservation ----- Context ----- Replication ----- State	Interoperation	Category
	Annotation	1. receiveSIP
	Notification	2. reviewSIP
	Application	3. quarantineSIP
	Transformation	4. appraiseSIP
	Search	5. prepareAIP
	Index	6. reviewAIP
	Ingest	7. storeAIP
	Interpretation	8. provideDIP
	Characterization	9. monitorPreservation
Inventory		
Replication		
Replication		
Fixity		
Storage		
Identity		

Category	Micro-Service
1. receiveSIP	verifyChecksum
2. reviewSIP	extractPackage assignIdentifier parseManifest cleanFilename
3. quarantineSIP	lockAccess virusCheck
4. appraiseSIP	identifyFormat validateFormat extractMetadata decidePreservationAction
5. prepareAIP	gatherMetadata normalizeFiles createPackage
6. reviewAIP	decideStorageAction
7. storeAIP	writePackage replicatePackage auditFixity readPackage updatePackage
8. provideDIP	uploadPackage updateMetadata
9. monitorPreservation	updatePolicy migrateFormat

Table 1. Archivemata micro-services

Figure 4-4: Comparison of Merritt vs. Archivemata micro-services for Pre-Ingest.

The workflow model follows the BPMN 2.0 notation and has been created using Eclipse-based toolsets. The pool is divided into swim lanes each covering an actor’s role during Pre-Ingest, i.e. Producer, OAIS Activity, Archivist, Archivemata micro-services. Each process is described by a corresponding use case. The Archivemata Use Cases are well documented and most important guidelines for the implementation of the Pre-Ingest tool. The micro-services are used to implement required functionality such as the minting of a unique identifier, virus scanning, executing scripts, launching the transformer.

4.2.1.1 Submission of Information Packages

The notification contains information about the structure of to-be submitted content and metadata. Producers (often referred to as ‘content providers’ in the BHL-Europe context) inform the Archivists about their intent to submit information packages. In the following steps the Archivist validates that the structure of the submitted content is OK and informs the producer about the outcome. Producers submit information packages by uploading their content and metadata to the NHM servers. Pre-Ingest provides the necessary means to

⁶ We needed to include the workflow diagram in two parts so that all steps are visible and readable.

producers to be able to submit SIPs: Pre-Ingest File Submission Guidelines, access to the storage cluster by SFTP and WebDAV. Technical notes (see Appendix III.II) have been created for both functionalities and provided to the relevant target groups. The technical notes are attached to this document for further reading.

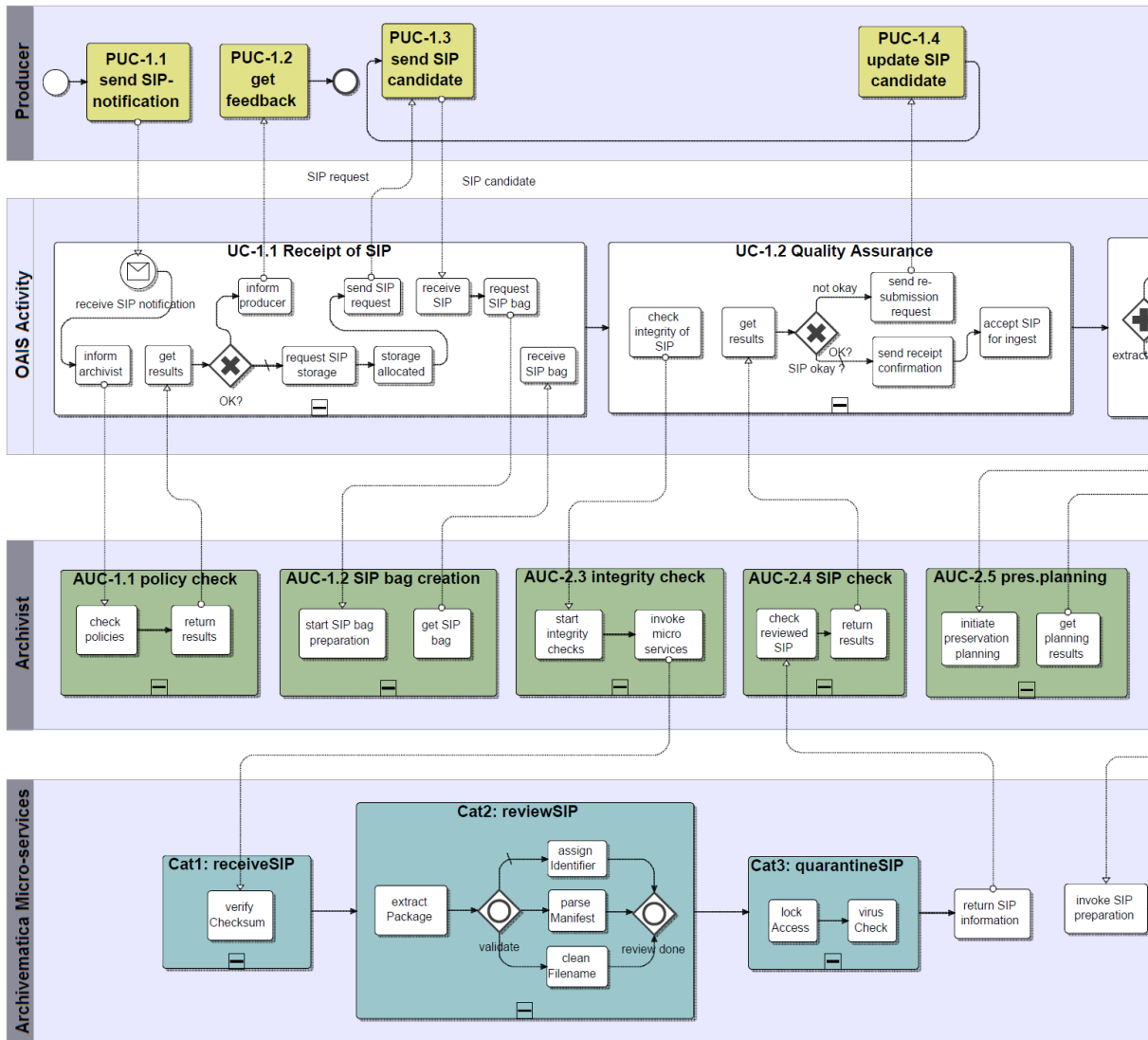


Figure 4-5: Pre-Ingest Workflow showing pool with swim lanes / part 1.

4.2.1.2 Processing of submitted Information Packages

Producers can't always follow the Pre-Ingest File Submission Guidelines as close as 100%. Some producers already started creating their content and metadata before the guidelines have been published and don't have the resources to adapt their data to the guidelines. Pre-processing of submitted data is required to make it even possible to work with it. Data harmonization during Pre-Ingest is one of the most time and resource consuming work therefore, but necessary. The required processing steps closely follow the Archivemata micro-services, i.e. extracting the package information, assigning identifiers, calculating file hash sums, checking for viruses, identification of package contents (titles, items), extraction of technical metadata, optical character recognition, semantic data enrichment.

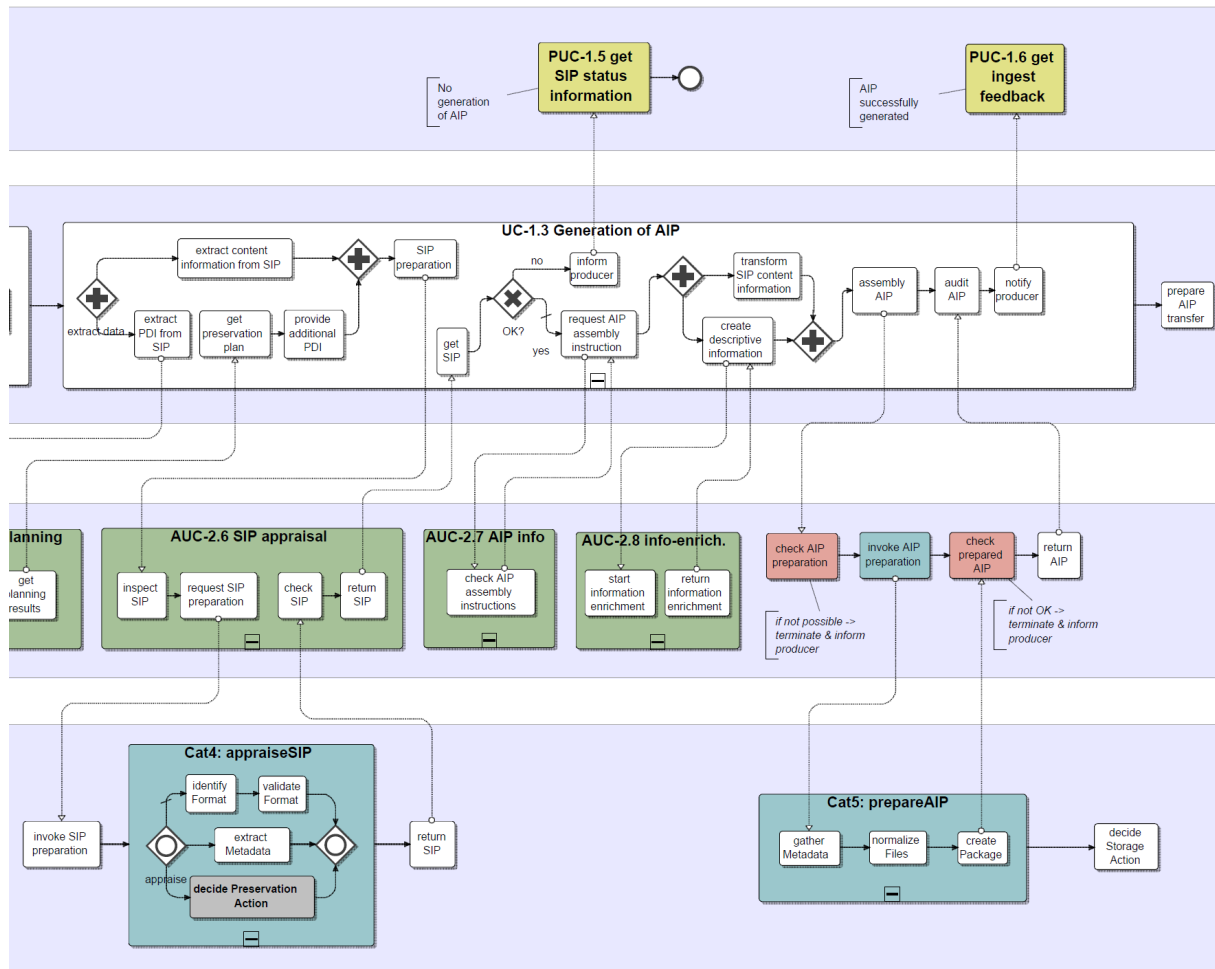


Figure 4-6: Pre-Ingest Workflow showing pool with swimlanes / part 2.

4.2.1.3 Preparation of Archival Information Package

The AIP is prepared for ingest at the end of the Pre-Ingest workflow. The generated METS file including all submitted and generated metadata and contents are moved onto the NFS storage for directory ingest.

4.2.2 Pre-Ingest in context of the OAIS function model

The Pre-Ingest component isn't part of the official 'blue book' OAIS standard⁷, but a well sought-after component in most archival systems. Examples include Apache OODT which is developed by NASA, or The Fascinator 2 which is developed by the University of Southern Queensland. Pre-Ingest is necessary to implement functionalities such as data harmonization which are missing in the official OAIS standard. Pre-Ingest happens right before Ingest but also covers aspects of Ingest. This is important to point out since also parts of this component description will refer to functionalities seen and covered usually during ingest (see Figure 4-7, Figure 4-8).

⁷ http://nssdc.gsfc.nasa.gov/nost/isoas/ref_model.html

OAIS function model

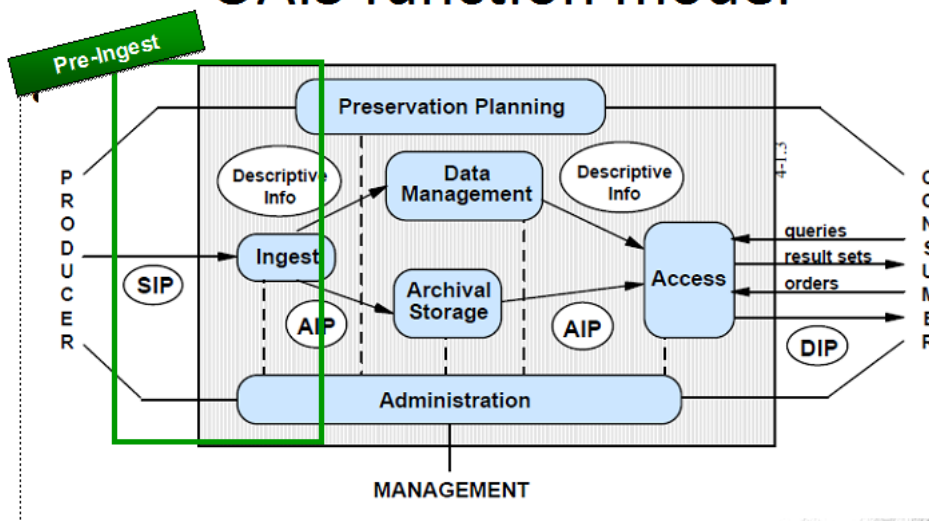


Figure 4-7: OAIS function model showing the location of Pre-Ingest.

The OAIS function model foresees SIPs to be ingested into the Ingest component. Pre-Ingest extends Ingest by functionalities to prepare SIPs and to successfully transform SIPs into AIPs. These AIPs are ingested into the Archival Storage system unchanged.

OAIS Functions of Ingest

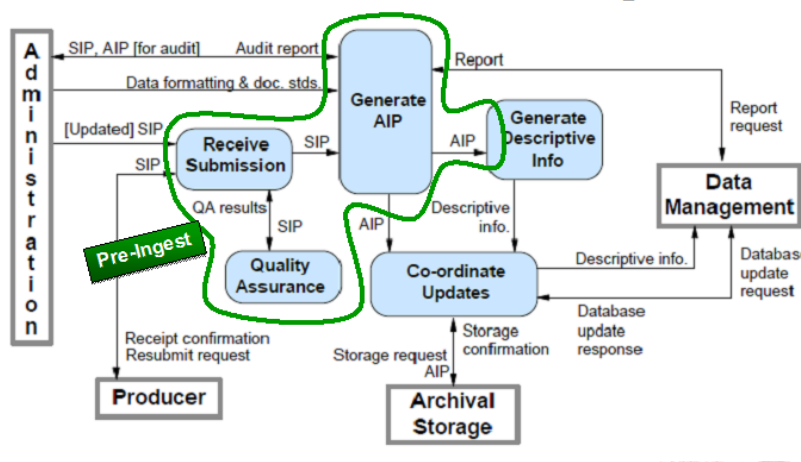


Figure 4-8: OAIS functions of ingest showing which areas are also partly covered by the Pre-Ingest component.

4.2.3 Data Harmonization

It is necessary to transform submitted information packages into a common structure so that further processing can happen. Since content providers submit very differently structured SIPs it's important to harmonize the content and metadata. We are using METS as a container format for transporting the different metadata formats and linking them to the content, i.e. the files. The Pre-Ingest Tool is developed to facilitate this process. Various requirements need to be fulfilled so that data can be automatically harmonized. Content providers should adhere to the Pre-Ingest File Submission Guidelines (see Appendix III.I) as closely as they can. Any

exceptions need to be manually handled, and additional preprocessing steps are necessary to transform SIPs so that they can be handled by the Pre-Ingest Tool.

Metadata mapping is also a part of data harmonization. Bibliographic metadata of content providers has been mapped to MODS inside of OLEF. Metadata mapping has been discussed based upon the documentation of the metadata data flow in the German prototype (see Appendix III.IV). A metadata gateway has been introduced as conclusion, so that metadata mapping happens before Pre-Ingest inside of the BHL-Europe architecture.

4.2.3.1 Pre-Ingest Tool

The tool facilitates the generation and control of submitted content and metadata. The deployment diagram for Pre-Ingest shows the various components which are deployed and used to facilitate incoming data (see Figure 4-11). It's actually comprised of three different components:

- Web-application for user interaction, extraction and transformation of information packages, integration with external and custom developed web-services and micro-services
- SFTP / WebDAV access to the BHL-Europe storage
- Scripting environment to implement individual processing steps

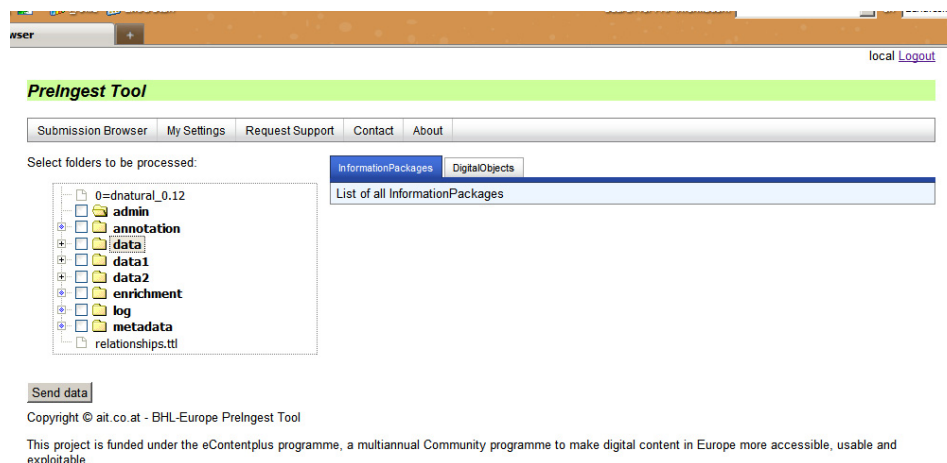


Figure 4-9: Web application for the Pre-Ingest Tool. Submitted directories can be selected for further processing.

The web application looks into each content provider's data and starts the data harmonization process for selected directories. The tool integrates various web services which have been built to interface different micro-services as specified in the workflow, i.e. schema mapping tool, ClamAV⁸ anti-virus daemon, NOID minter⁹. It was developed using a best-practice approach for implementing loosely coupled, highly available, well performing, scalable web

⁸ <http://www.clamav.net/lang/en/>

⁹ <http://search.cpan.org/dist/NoId/>, also see Appendix III.III about BHL-Europe GUID which is based on the NOID minter adaptor.

applications. Therefore we decided to use parts of the Java Spring framework¹⁰. The Spring Integration¹¹ framework is used on top of it to implement the necessary messaging system to pass digital objects (scans, metadata, ...) from one processing step to the next (see Figure 4-10). Digital Objects are passed from one channel to the next and service activators in between operate on the data objects which are passing through.

¹⁰ <http://www.springsource.com/>

¹¹ <http://www.springsource.org/spring-integration>

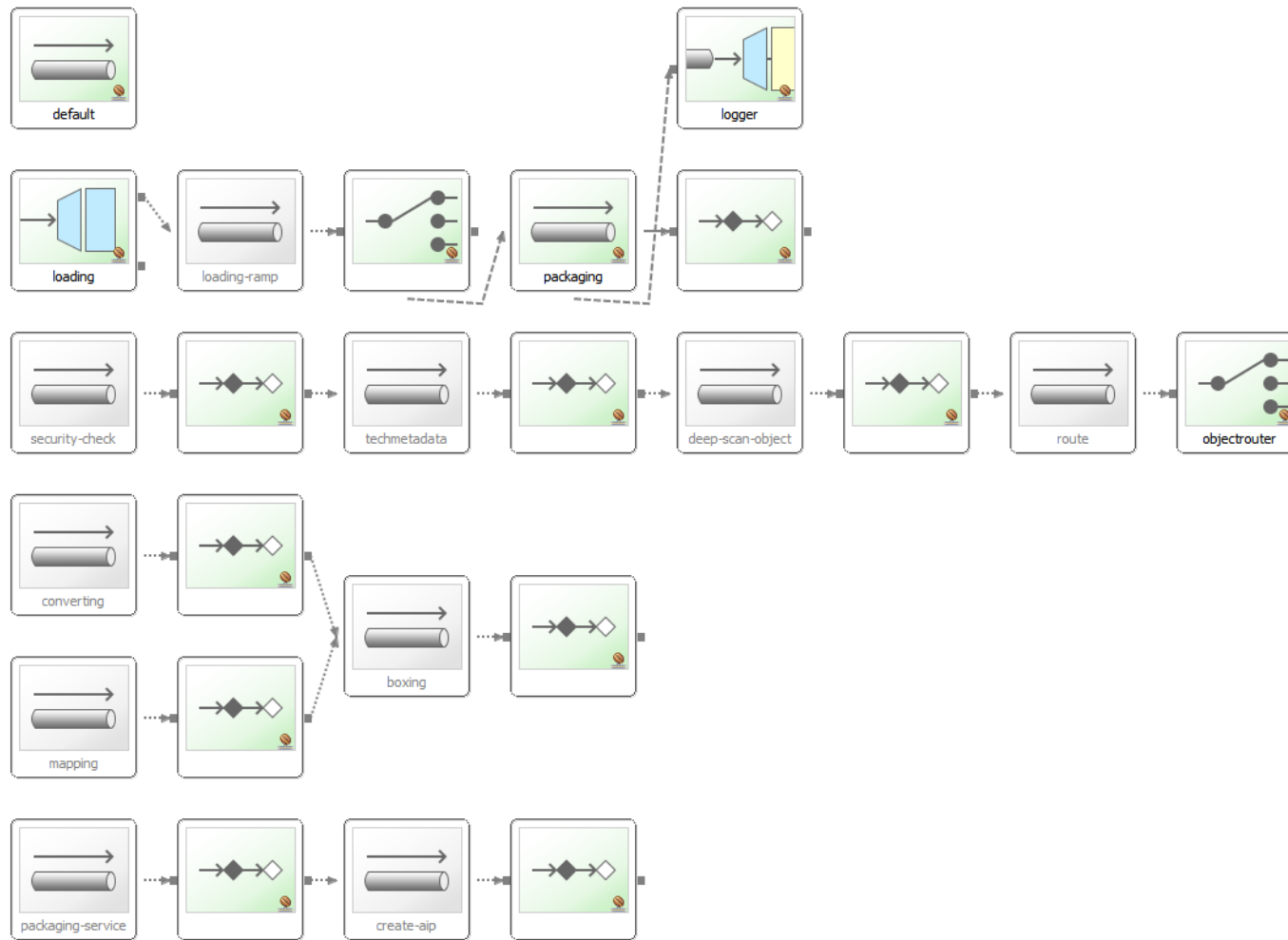


Figure 4-10: Visualization of different messaging channels and service activators (such as loading-ramp, security-check, boxing, create-aip, ...) of Pre-Ingest Tool. Starting from loading gateway in the upper left until create-aip in the lower right.

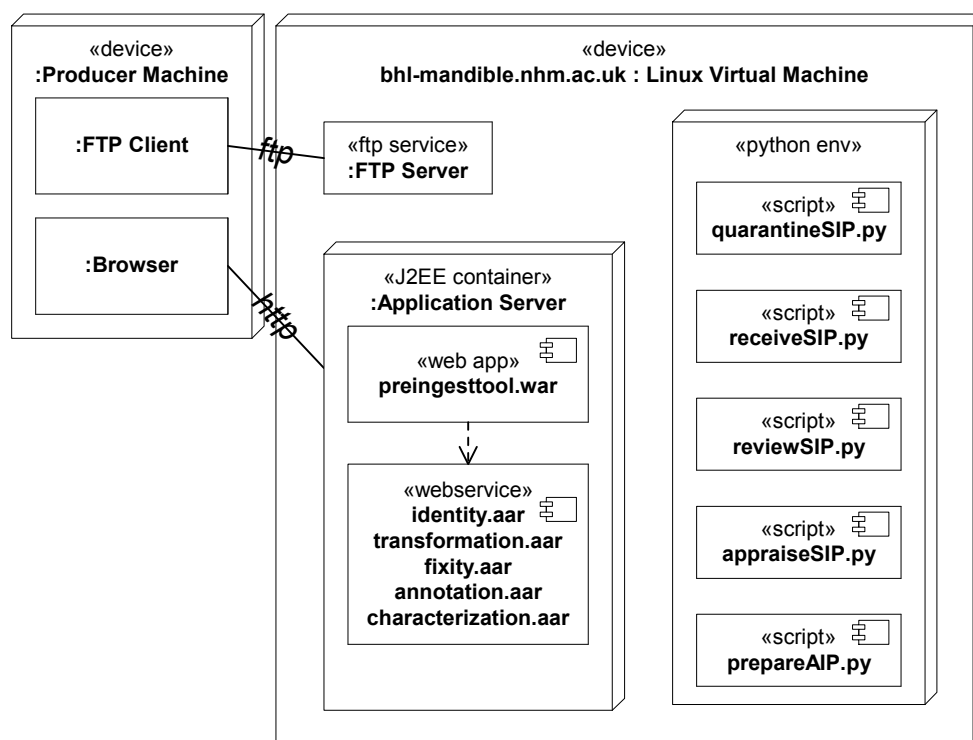


Figure 4-11: Deployment diagram of the Pre-Ingest tool.

It is important to point out that the web application handles any transformation and extraction of content and metadata and marshals them together into METS.

The different scripts in the scripting environment are necessary to cover the individual preprocessing. We decided to implement the preprocessing using dynamic scripting languages which significantly speeds up development and can be easily adapted. We are also using various adapted scripts to process content and metadata to preview AIPs. We use these previews of items to ingest content into Europeana as long as the BHL-Europe archive isn't yet finished.

The FTP server has been integrated as an FTPS server due to security regulations at NHM London's site. It proved difficult for several content providers to upload data using FTPS due to firewall issues with the specific protocol. Therefore a WebDAV server has been configured on top of Apache using mod_dav.

The source code has been checked into the bhl-bits repository on Google code¹², and should provide the ideal basis for further development and continuous development, improvement, integration.

4.2.4 Metadata Enrichment

Pre-Ingest also covers necessary data enrichment which is stored inside of METS following the BHL-Europe Schema OLEF (Open Library Exchange Format, as defined by NHM

¹² <http://code.google.com/p/bhl-bits/source/browse/#svn%2Ftrunk%2Fbhl-e%2Fpre-ingest>

Vienna). We are using JHOVE¹³ 1.6 to extract technical metadata from the images and also store them inside METS. This is an important step to have standard conformant metadata for preservation purposes. The most important step is the optical character recognition (OCR) of scanned pages. This is the basis for any further taxonomic intelligence processing since we are adding all text and all identified taxa to the Metadata. OCR is implemented using the software tesseract¹⁴ which leaves room for improvement. Since autumn 2010 tesseract is available as version 3.00 and supports new ways of customizing character recognition. First tests with BHL-Europe scanned images show good results and therefore version 3.00 is integrated into the BHL-Europe Pre-Ingest workflow. Experimental support for hOCR¹⁵ is also built into tesseract 3.00 which outputs text including all words' bounding boxes. Therefore a straight-forward integration with IA BookReader's full-text search seems possible. Other options have been ruled out due to the high cost they are associated with (i.e. ABBYY¹⁶). Taxon finder and uBio webservices¹⁶ are used to extract taxa out of the OCRed text.

BHL-Europe is not only interested in just implementing existing OCR technologies but also to improve these technologies. OCR errors are a major problem for taxonomic intelligence technologies as the data for enrichment are based on the OCR text of digitised page images. Thus, improving OCR will improve the name finding and subsequently the search for taxonomic information in the digital biodiversity heritage literature. The improvement of OCR technologies is really a challenge for BHL-Europe, being only one task among many other tasks. Therefore, we are collaborating with the EU-funded IMPACT project (Improving Access to Text). Recently we agreed to be involved in the test of the applications and services developed within the IMPACT project. A test set is in progress to be submitted to IMPACT. Testing and evaluation will happen over the summer. The goal of this collaboration is to find out to what extent the IMPACT approach can help to improve the OCR of BHL-Europe content.

It's important to point out that web services from VIAF and Species2000 aren't used for metadata enrichment, but are used for search term amplification at the Portal. In case metadata enrichment with Species2000 is required, Pre-Ingest can easily be adapted to interface with the Species2000 web service.

We are using Pentaho Kettle¹⁷ and XSL-based technologies to facilitate the enrichment (see Figure 4-12) of data during the Pre-Ingest Workflow. Pentaho Kettle is an Open-Source state-of-the-art ETL (Extract, Transform, Load) Tool.

¹³ <http://hul.harvard.edu/jhove/>

¹⁴ <http://code.google.com/p/tesseract-ocr/>

¹⁵ https://docs.google.com/View?docid=dfxcv4vc_67g844kf

¹⁶ <http://www.abbyy.de/>

¹⁷ <http://kettle.pentaho.com/>

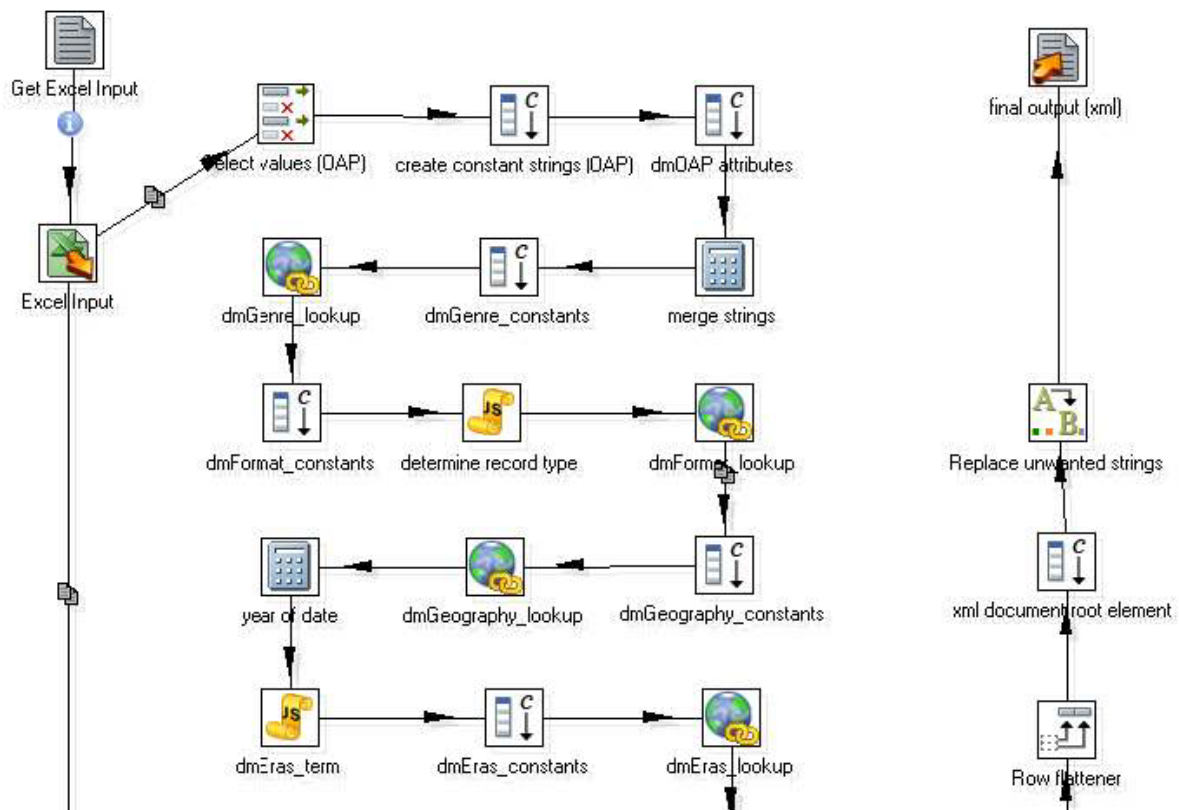


Figure 4-12: Pentaho Kettle transformation from Excel to XML using multiple enrichment steps in between.

4.3 Ingest

“This component provides the services and functions to accept Submission Information Packages (SIPs) from the PreIngest module or Producers directly (or from internal elements under Administration control) and prepares the contents for storage and management within the archive. Ingest functions include receiving SIPs, performing quality assurance on SIPs, generating an Archival Information Package (AIP) which complies with the archive’s data formatting and documentation standards, extracting Descriptive Information from the AIPs for inclusion in the archive database, and coordinating updates to Archival Storage and Data Management.” [D3.4, p.25]

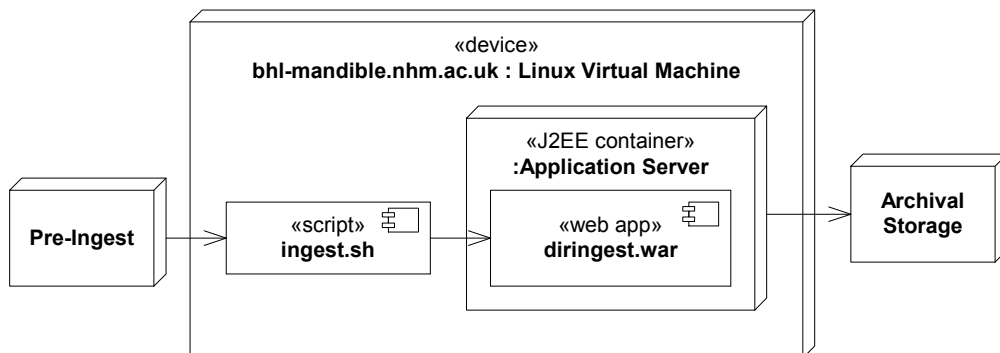


Figure 4-13: Deployment diagram of Ingest component showing the script used for batch ingest of AIPs and the web application Directory Ingest Service (in the context of Fedora Commons).

The output from Pre-Ingest is an AIP which is ready to be ingested into the Archival Storage system (see Figure 4-6, step “prepare AIP transfer”). This transfer is the only process covered by the Ingest component (see Figure 4-14, Figure 4-13). All required steps to generate AIPs have been covered during the Pre-Ingest workflow (see Figure 4-8).

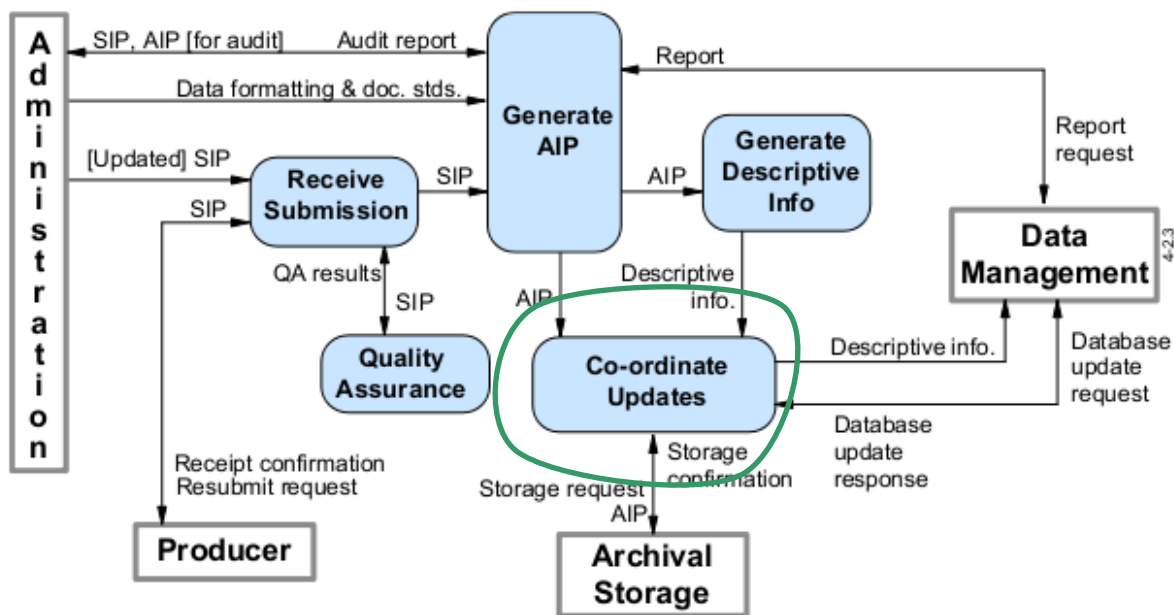


Figure 4-14: The ingest component covers the process required to ingest AIPs into the Archival storage system.

The coordination of updates as seen in Figure 4-14 happens by the means of an ingest script. This also means that the implementation of this specific OAIS functionality is done based on the requirements of the Archival Storage system. Fedora Commons is used for Archival purposes and therefore the script used to ingest AIPs into Fedora Commons is based on the directory ingest scripts already available. The scripts are adapted to facilitate putting AIPs into Fedora Commons. The binding element between the Pre-Ingest, Ingest, Archival Storage components is the minted identifier during the Pre-Ingest workflow. The ingest mustn't be a one way dataflow – AIPs must also be able to be transferred back into the Ingest component in case further processing is needed before ingestion. Therefore this script also makes it

possible to simply request and download data from Archival Storage based upon the identifiers. This identifier would also be compatible with LOCKSS which has been investigated to create the distributed storage of the archive (see Appendix III.V).

The ingest architecture as envisioned in D3.4 needs to be reasonably scaled down for the above described processes since no other logic is really necessary in the context of BHL-Europe. All interaction happens based on access to the NFS share which holds the content provider’s submission. Pre-Ingest provides AIPs by copying them over to the NFS share, each directory is named after the minted identifier, and waiting to be pulled into Archival Storage by the directory ingest script. This delivers the same functionality as described in Figure 4-15, but proves to be vastly simplified and therefore more flexible and easier to deliver. Automated communication with Administration and Data Management components has been moved over to Archival Storage since Fedora Commons provides the ideal utilities for these OAIS functionalities partially out of the box.

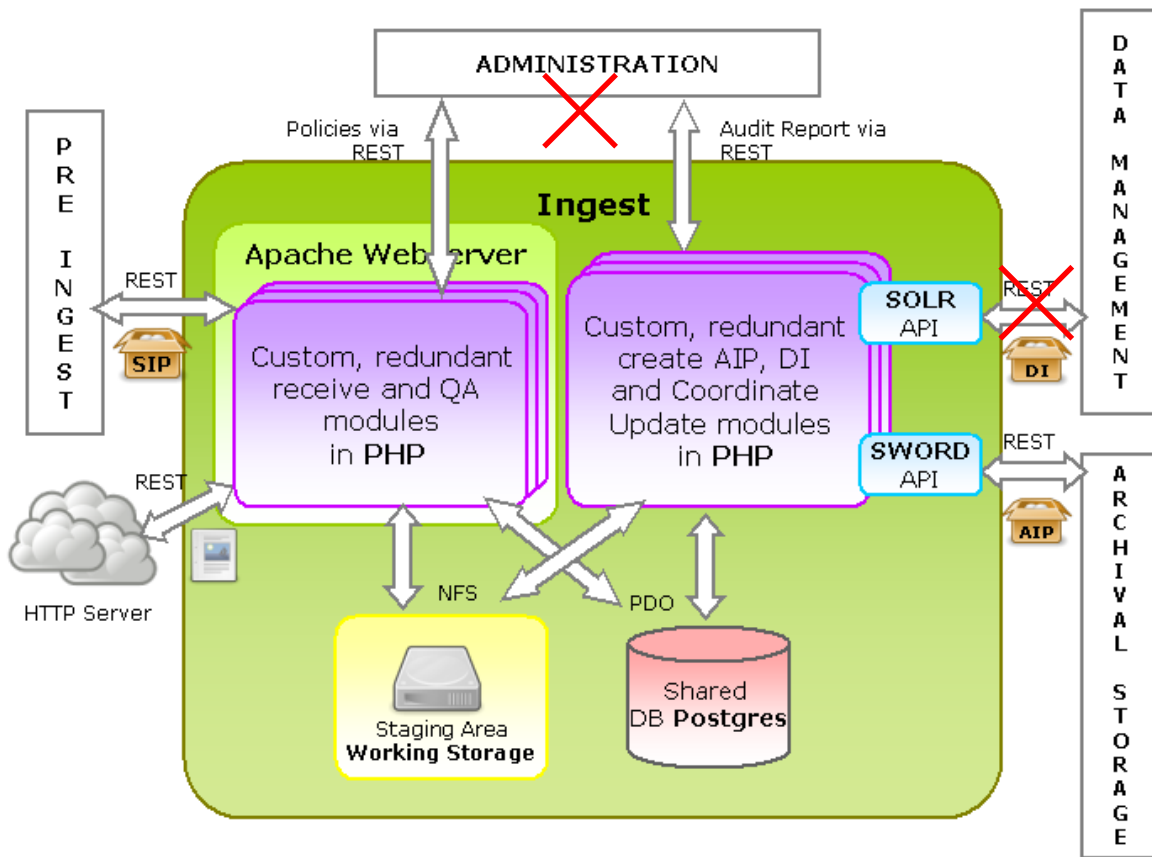


Figure 4-15: Ingest architecture as documented in Deliverable 3.4.

4.4 Archival Storage

This component provides the services and functions for the storage, maintenance and retrieval of AIPs. The work on Archival Storage is mainly composed of the installation, configuration, and integration of the open source components Tomcat, MySQL, and Fedora Commons with the infrastructure of BHL-Europe housed at the Natural History Museum, London. As such documentation of Fedora Commons is provided as an appendix.

The differences between the current version and D3.4 are:

- The SWORD interface and plugin for Fedora is not used. At this point the added value of providing a SWORD interface is not known and thus does not seem worth implementing currently. A SWORD interface can always be added in parallel at a future date if the requirement arises.
- The MySQL database is used instead of Postgresql. The majority of the development team members were already familiar with MySQL thus making development faster.
- A small custom extension of the underlying lowlevel storage based on Akubra module has been written to accommodate the different types of storage mainly short term(disk based) and long term(disk and tape based).

The following diagram shows the main software modules used to implement Archival Storage.

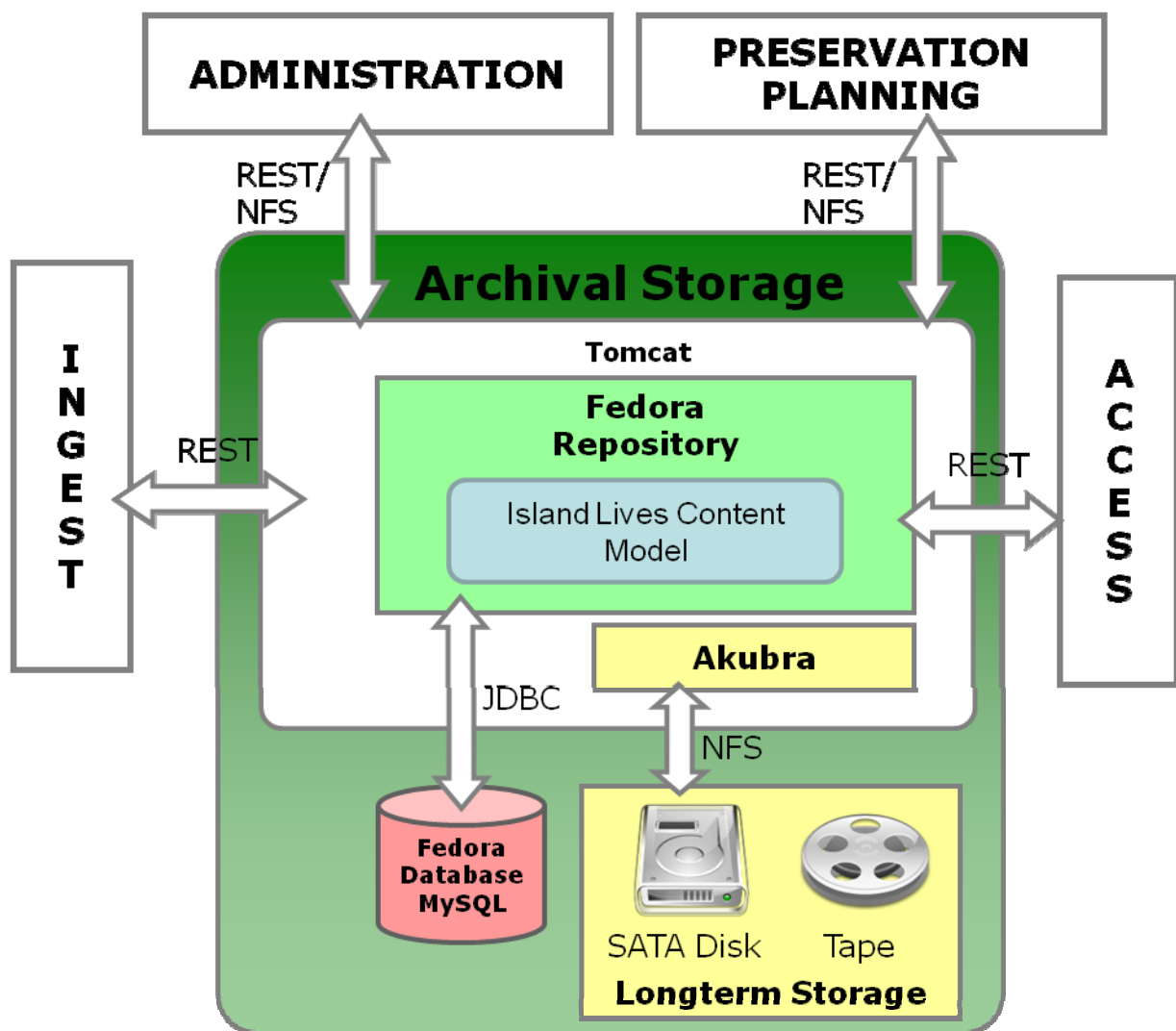


Figure 4-16: Archival Storage Architecture

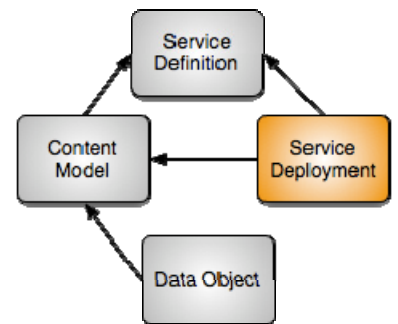
4.4.1 The Fedora Digital Object

Fedora defines a generic digital object model that can be used to persist and deliver the essential characteristics for many kinds of digital content including documents, images, electronic books, multi-media learning objects, datasets, metadata and many others. This digital object model is a fundamental building block of the Content Model Architecture and all other Fedora-provided functionality. The basic components of each digital object are:

- PID: A persistent, unique identifier for the object.
- Object Properties: A set of system-defined descriptive properties that are necessary to manage and track the object in the repository.
- DataStream(s): The element in a Fedora digital object that represents a content item.

There are four distinct types of Fedora digital objects that can be stored in a Fedora repository. The distinction between these four types is fundamental to how the Fedora repository system works.

- Data Object: used to store digital content entities
- Service Definition Object: used to store service descriptions
- Service Deployment Object: used to deploy services
- Content Model Object: objects used to organize other objects



4.4.2 Islandora Content Model

For BHL-Europe we use the “**Islandora content model**” to determine what objects will be ingested and how the object will be managed on ingest.

The Islandora Content Model extends the Fedora Content Model Architecture (CMA). Specifically we use the “**bookCModel**” and the “**pageCModel**”. These two objects are provided by the Island Lives drupal module (Islandora Book) and can be ingested via the drupal admin interface.

Each digital object using the bookCModel or pageCModel will represent a book or a page. So it is easier to create relations between a book and its pages.

4.4.3 Deployment

The following diagram shows the deployment diagram for Archival Storage.

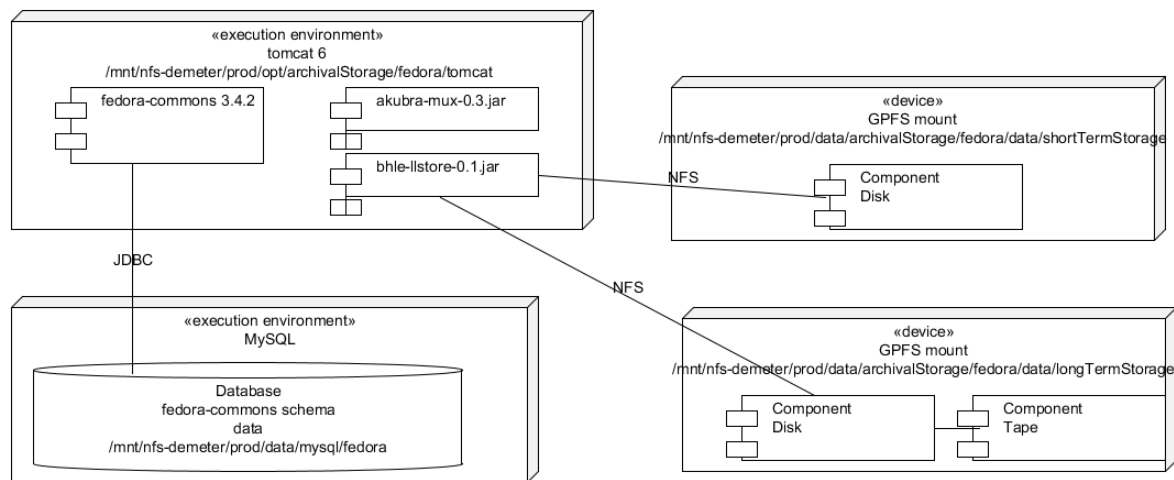


Figure 4-17: Archival Storage Deployment Diagram

4.5 Data Management

Data Management provides services and functions for populating, maintaining, and accessing Descriptive Information which identifies and documents archive holdings and administrative data used to manage the archive. The work on Data Management is mainly composed of the installation, configuration, and integration of the open source components Apache, Drupal 7, Islandora Drupal Module, MySQL, and Fedora Commons with the infrastructure of BHL-Europe housed at the Natural History Museum, London. As such documentation of Islandora is provided as an appendix.

The differences between the current version and D3.4 are:

- The addition of Islandora to provide an graphical user interface to the Fedora repository instead of custom PHP code. Fedora provides a user interface, however, the usage is for advanced users such as developers and not necessarily functional administrators such as librarians or scientists. To fulfill this simpler UI requirement Islandora (<http://islandora.ca>) was evaluated and chosen. Islandora is deployed as a module to Drupal. In order to maintain loose coupling and independence between OAIS components a separate Drupal instance will be used for Data Management from the Portal.
- The “search” functionality was only required in the Portal and thus was moved from the Data Management component to the Portal component. The impact to Data Management is that the Solr search engine is no longer needed.
- The MySQL database is used instead of Postgresql. The majority of the development team members were already familiar with MySQL thus making development faster.

The following diagram shows the main software modules used to implement Data Management.

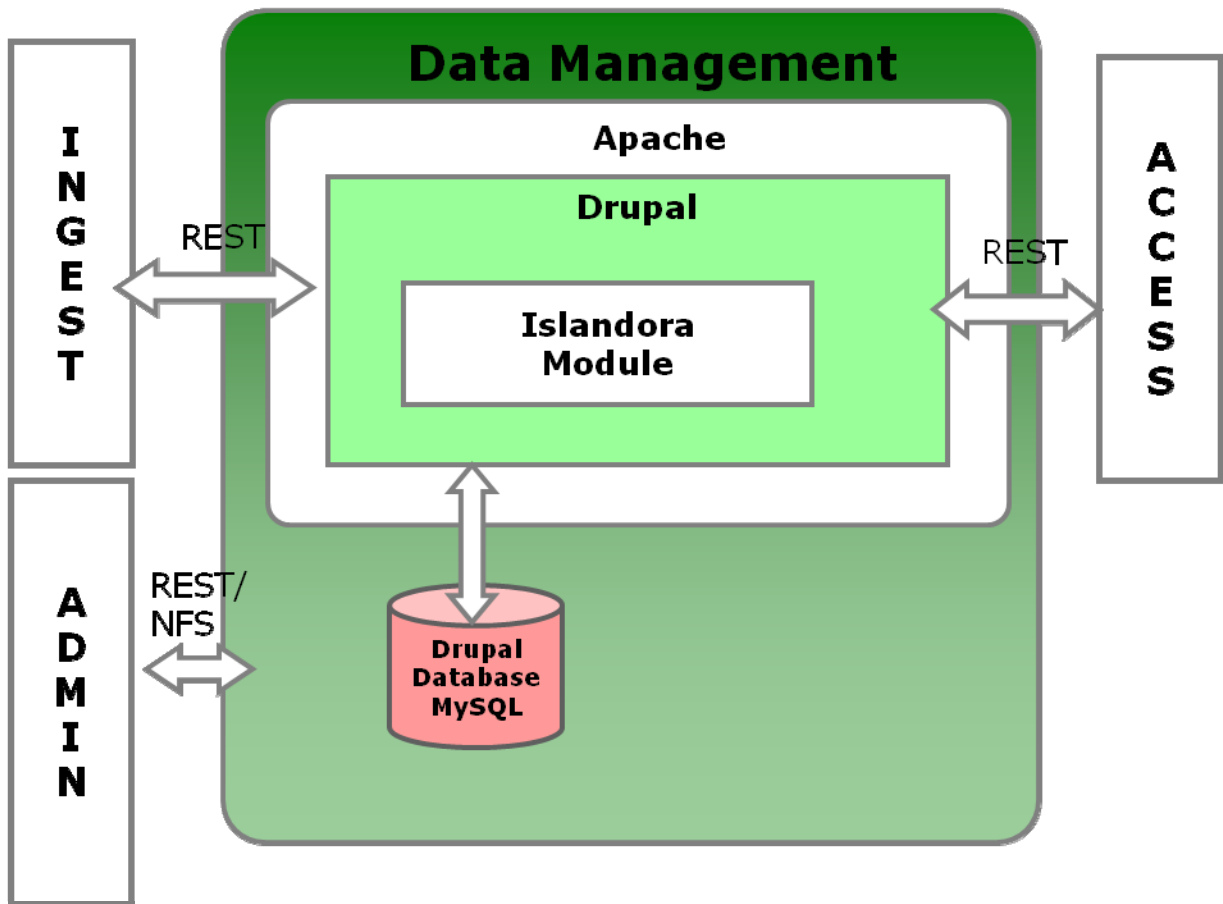


Figure 4-18: Data Management Architecture

4.5.1 Deployment

The following diagram shows the deployment diagram for Data Management.

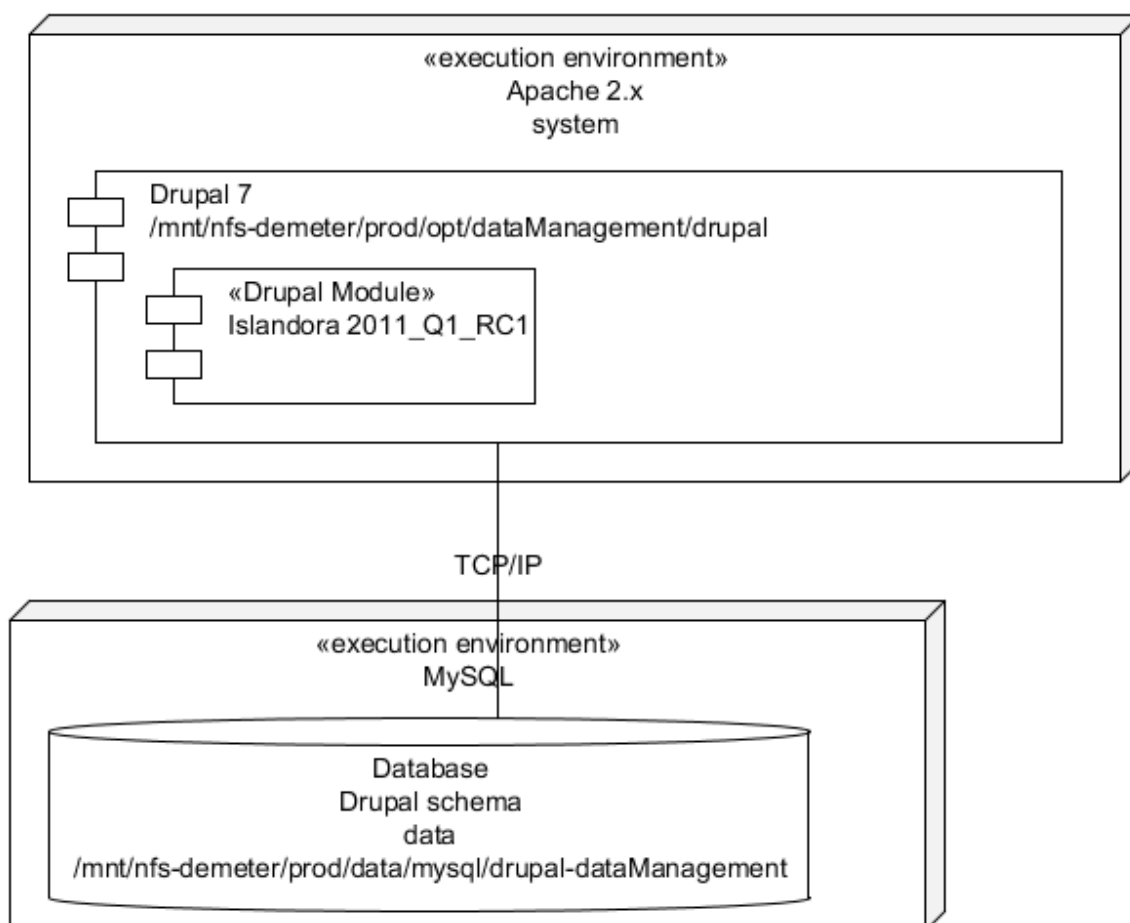


Figure 4-19: Data Management Deployment Diagram

4.6 Access

“This component provides the services and functions that support Consumers in determining the existence, description, location and availability of information stored in the OAIS, and allowing Consumers to request and receive information products. Access functions include communicating with Consumers to receive requests, applying controls to limit access to specially protected information, coordinating the execution of requests to successful completion, generating responses (Dissemination Information Packages, result sets, reports) and delivering the responses to Consumers.” [D3.4, p.p. 49-50]

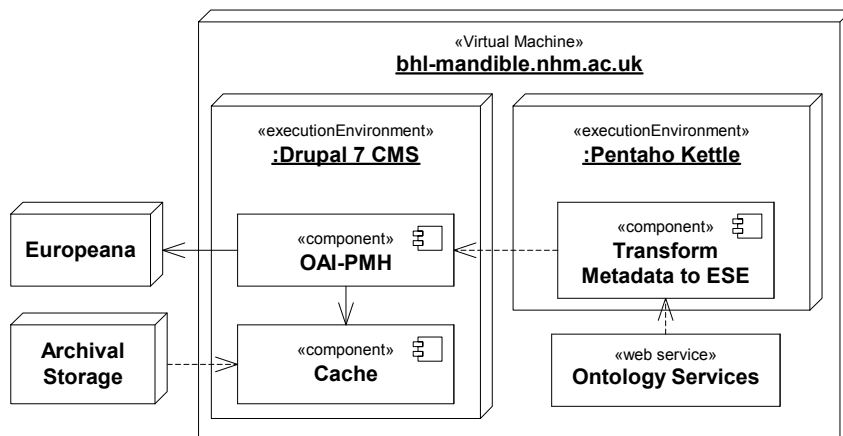


Figure 4-20: Deployment diagram of Access component.

The Access component (see Figure 4-20) has access to the Archival Storage component and might be understood as the simplified counterpart of the Ingest component. Whereas Ingest is responsible to put AIPs into the Archival Storage system, Access pulls them out of the Archival Storage system and pushes them into the Drupal portal. Thus the Access component transforms an AIP into a DIP (Dissemination Information Packages), and makes the content and metadata available for and on the Portal. A DIP is not only a visible BHL-Europe content node on side of the portal, but also ESE transformed MODS (coming out of OLEF), especially created for Europeanana. Again this process has been elegantly simplified and needs no attachment to other OAIS functionalities. A script is pulling data out of Fedora Commons using the well- documented HTTP interface and pushes metadata onto the Drupal portal (see 4.7.9 Drupal). The system-wide identifiers are used to have a unique reference between Access and Archival Storage.

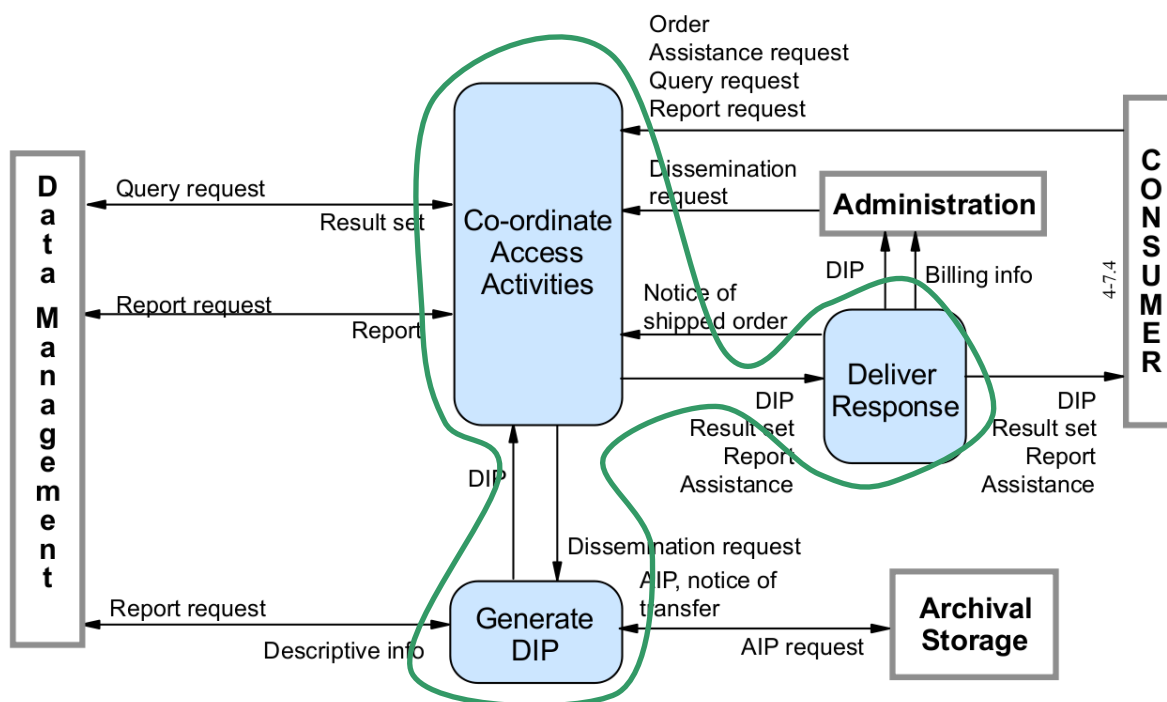


Figure 4-21: Access processes covered in the context of BHL-Europe.

The OAIS Access component has been adapted to the needs of the project and further toned. Specifically the communication with Data Management and Administration components has been moved over to the Archival Storage system. Of course the main functionalities remain the same. Data is being transformed by Pentaho Kettle jobs into ESE, stored in the Portal, and an OAI-PMH provider delivers the transformed ESE data to Europeana. The DIP cache has been moved over to the Portal where Drupal nodes are created which hold the metadata and unique identifiers.

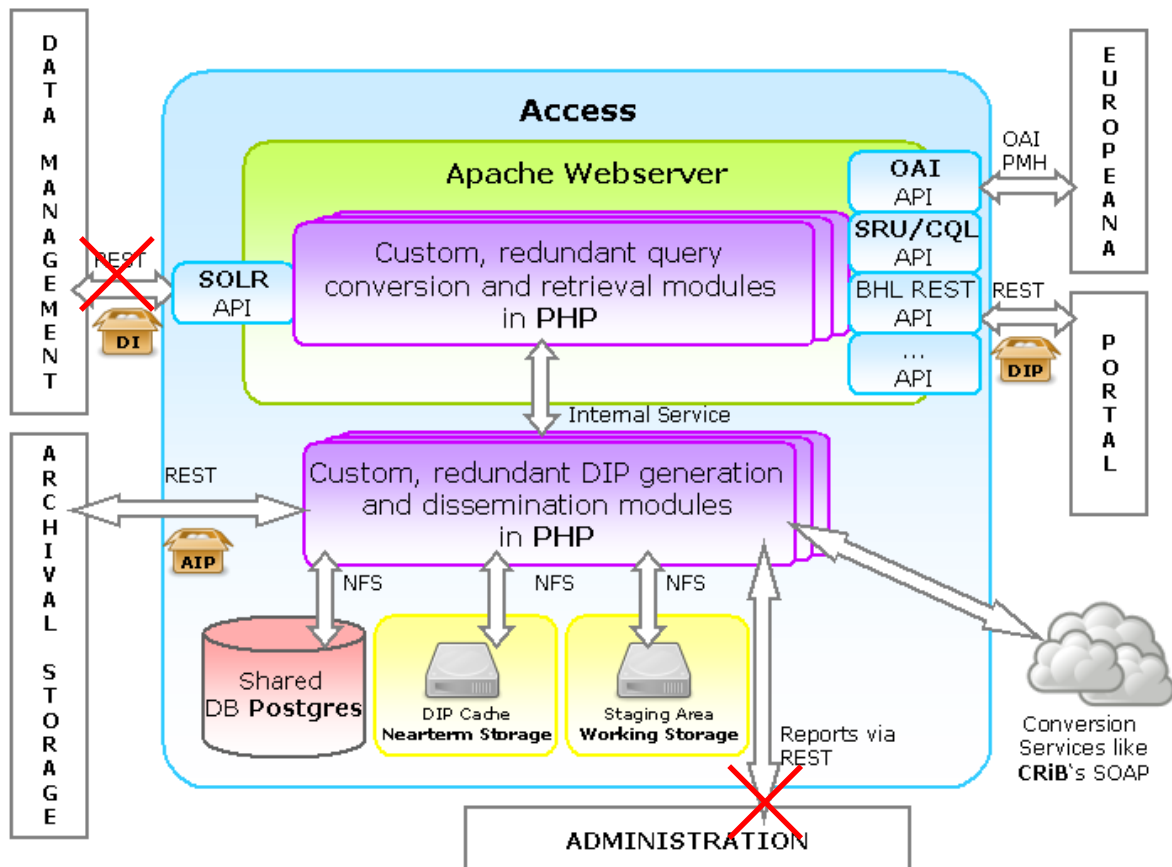


Figure 4-22: Access Architecture as planned in D3.4.

4.7 Portal

The BHL-Europe portal is developed based on the German prototype which has been demonstrated for D3.6. Functionality is ported to Drupal 7 modules (see Figure 4-23) which provide an ideal basis for further development. Core features have been identified and extracted out of the ‘Description of Work’ (DoW):

Feature Categories	Description
Simple Search.....	Describes how a simple Boolean search is initiated
Advanced Search.....	Describes how a more complex search is initiated
Browse Index.....	Describes by what navigational means content is browsed

User Profile Personalisation	Describes personalisation options for registered users
Search Results View.....	Describes how search results are displayed
Metadata View	Describes how metadata about titles/items is displayed
Content View.....	Describes how content (such as scanned images) is displayed
Multilingual UI.....	Describes that the portal needs to support multiple languages

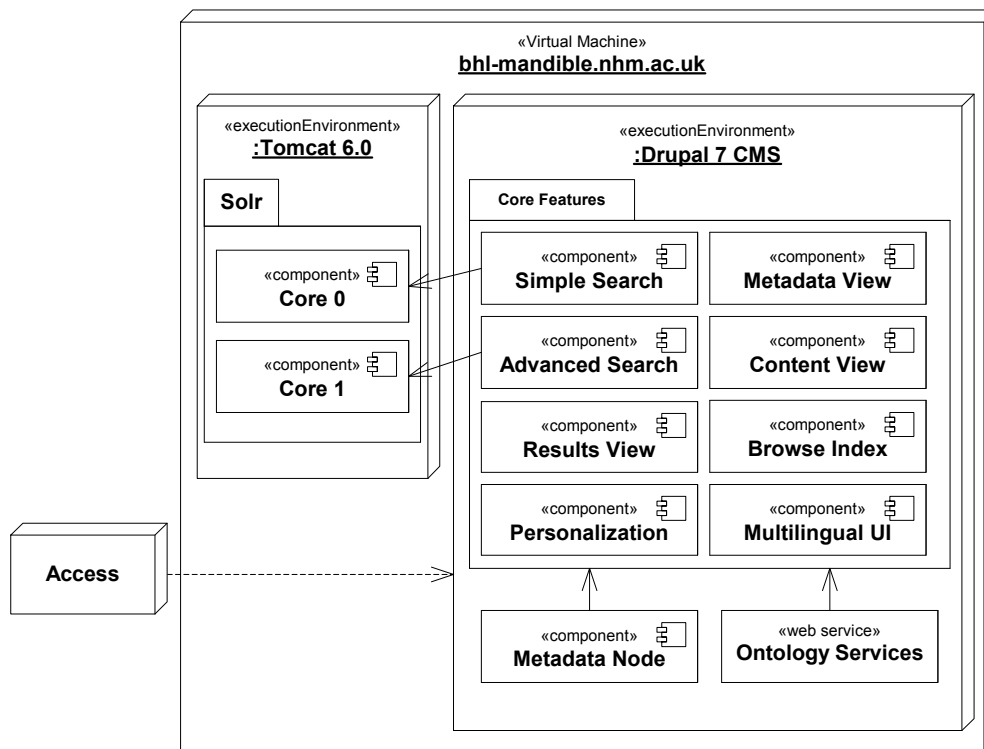


Figure 4-23: Deployment diagram showing Drupal 7 modules as components (core features and metadata node). Ontology services are used for taxonomic enrichment.

4.7.1 Simple Search

Simple Search provides the possibility to access basic search functionality with a single search field, that can be displayed on several positions of the portal. An auto-complete function helps users when entering a search term. The implementation of the simple search is based upon the Apache Solr Integration module¹⁸. This module indexes all contents of the portal system on a timely basis. Thus there are separate Solr cores (i.e. an index database) for simple search and advanced search.

¹⁸ <http://drupal.org/project/apachesolr>

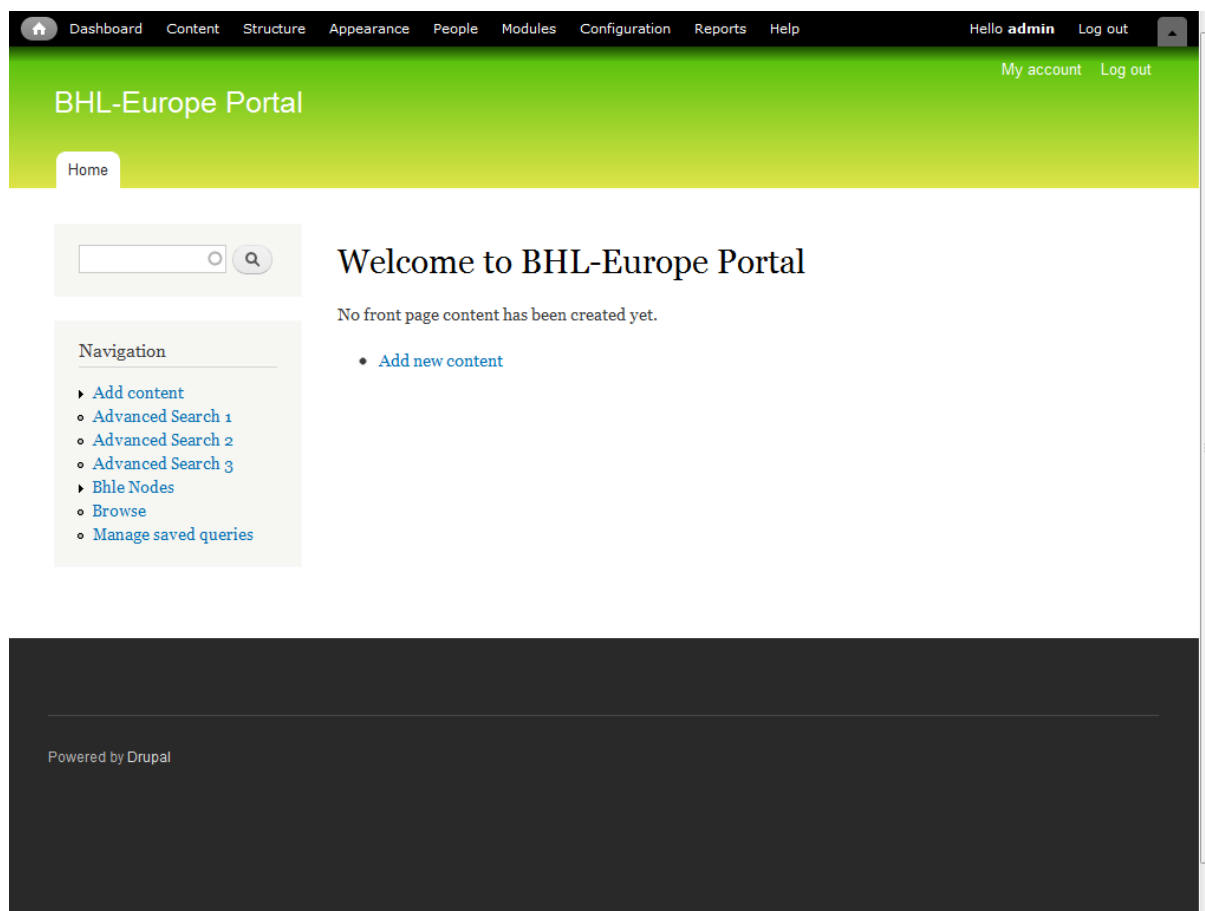


Figure 4-24: Simple search screenshot from current development version.

4.7.2 Advanced Search

The advanced search offers additional search adjustments. Various search masks are implemented. All development is based on the Drupal 7 (Forms) API, so interface design can happen independently from delivering core features and functionality. The core for advanced search is created manually based on the relevant fields contained in the metadata. Therefore it's possible to search for results while combining different search queries for different indexed metadata fields.

The first implementation of an advanced search mask is developed based upon the example of the Europeana advanced search mask where search fields can be specified and additional parameters can be added when needed. We decided to follow these examples since they are already used by users and represent a de-facto standard for advanced search query forms (see Figure 4-25, Figure 4-26). The second implementation of an advanced search mask is developed based upon the example of the Google advanced search mask. It enables users to make queries in the categories 'with all words', 'with exact phrase', 'with any word' and 'without words' (see Figure 4-27, Figure 4-28). The most flexible search is the advanced search, where users can use logical operators and other reserved words and parameters to specify the search query directly (see Figure 4-29).

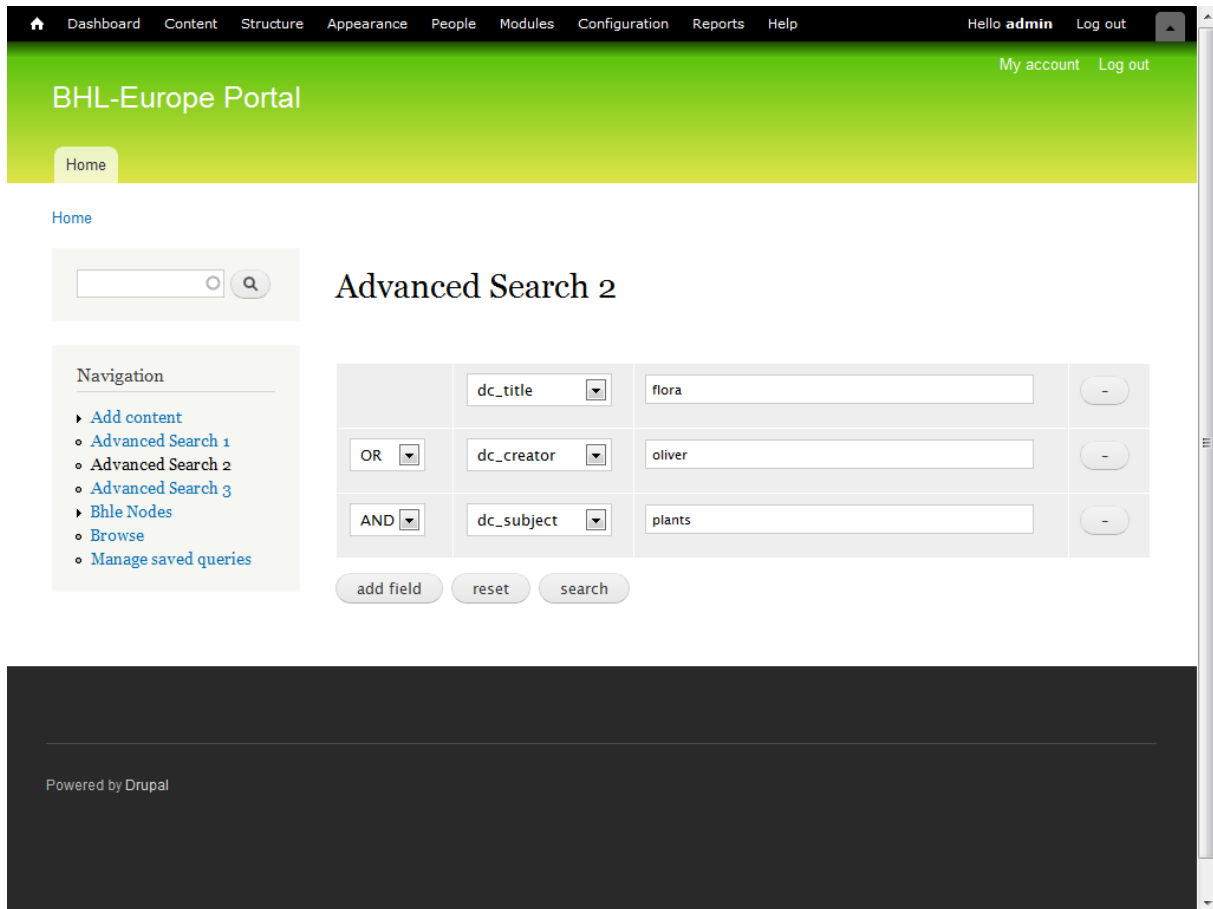


Figure 4-25: Advanced search from current development version. Multiple fields can be select.

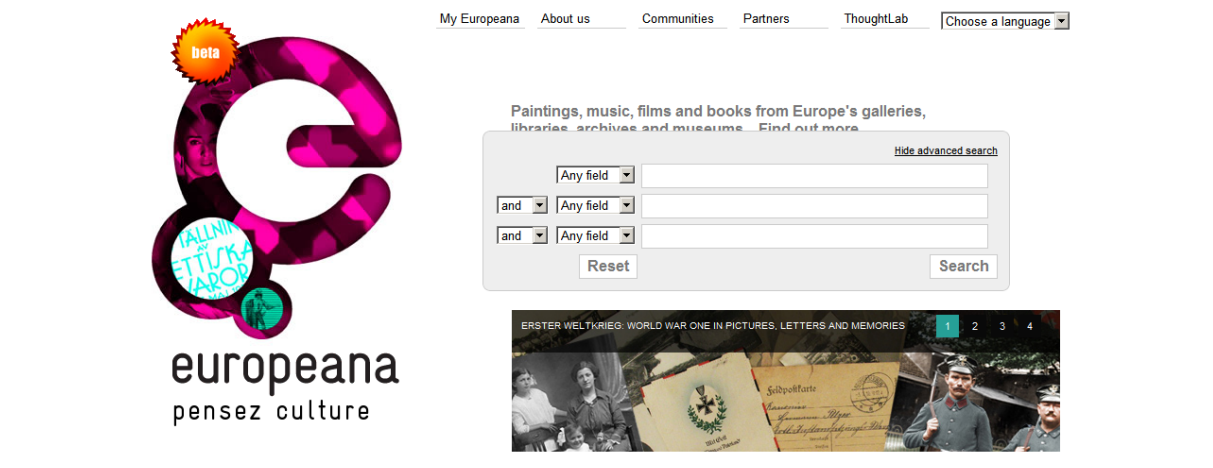


Figure 4-26: Advanced search example as seen on Europeana.

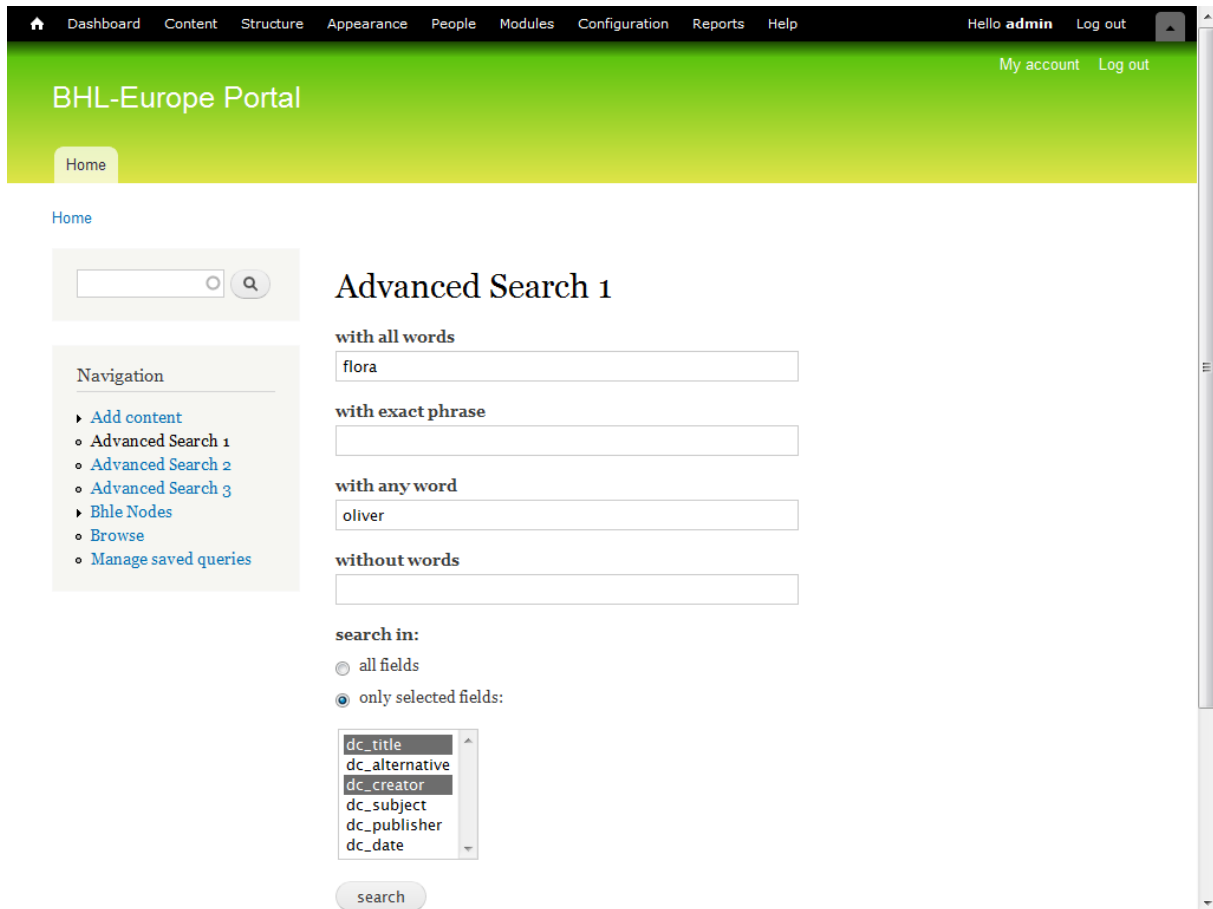


Figure 4-27: Advanced search (Google-like) from current development version.

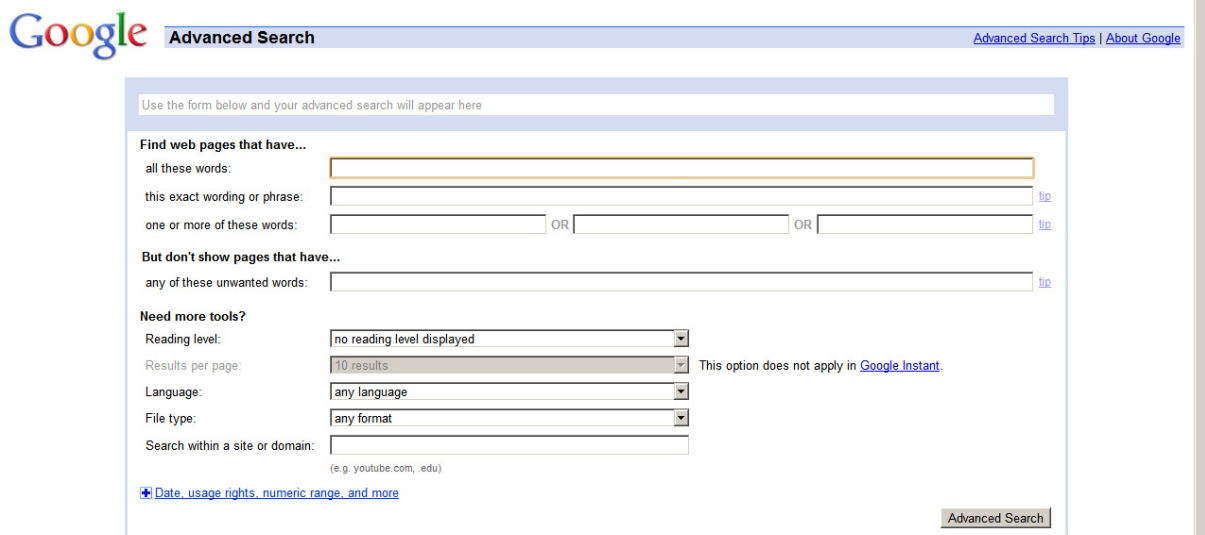


Figure 4-28: Advanced search as seen on www.google.com.

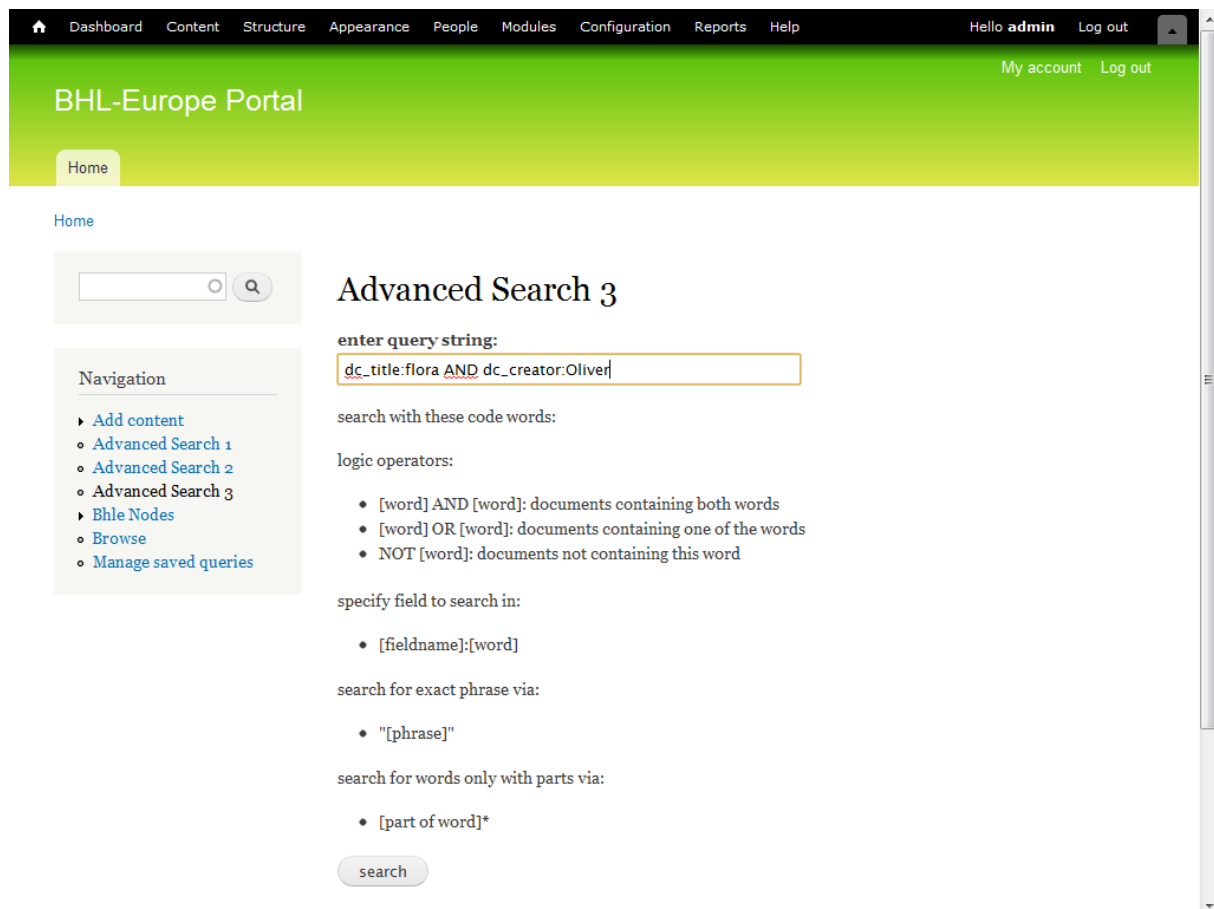
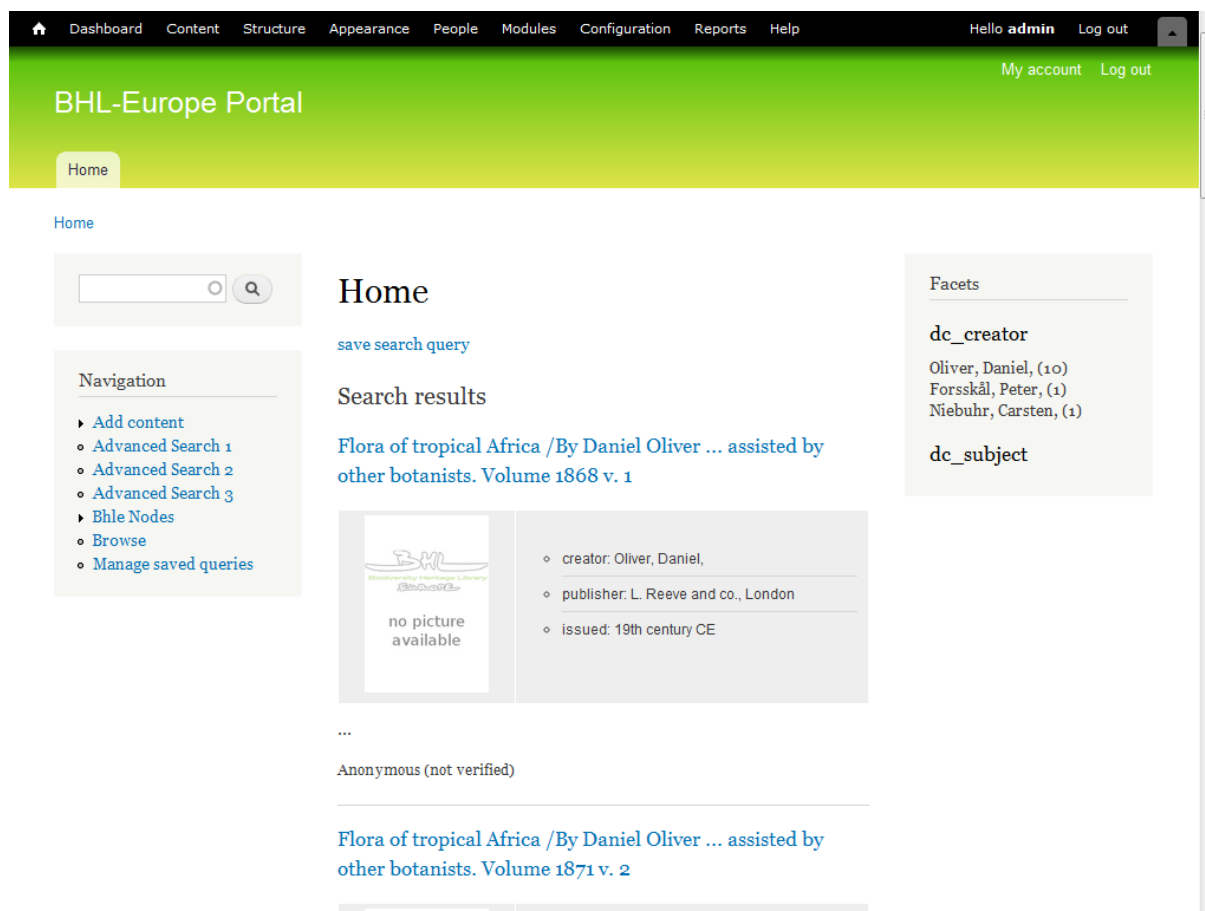


Figure 4-29: Advanced search screenshot from current development version. Search syntax can be manually entered and thus provides an optimised interface for experts.

4.7.3 Search Results View

Search results views need to be customizable to enable occasional and expert users alike to easily identify sought-after content. This means that users can change what and how search results are displayed using Extensible Stylesheet Language (XSL) by creating and uploading stylesheets on their own. Therefore multiple presentation formats, like a matrix of results, are possible (as seen on Europeana). The standard format is a simple listing of selected metadata fields with a fixed number of results. It contains title, creator, publisher, issued to implement the ‘who, when, where, what’ scheme. Controls for navigating to the next, previous and other pages are added. This pagination functionality is again based on Drupal 7 API.

Each record in the result-set is shown with title and a customizable stylesheet that displays parts of the metadata and possibly a thumbnail. Users are also capable of changing how data is displayed through the stylesheet definition.



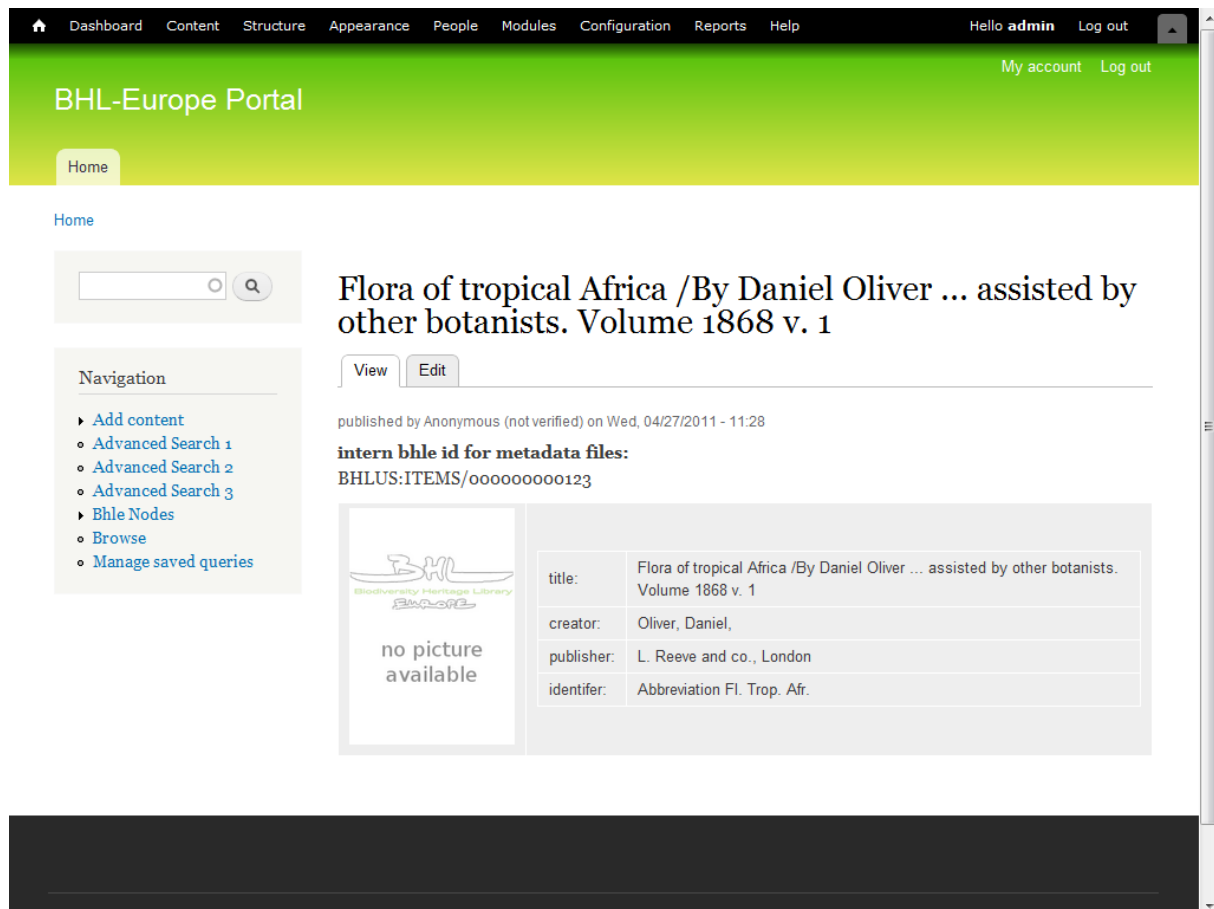
The screenshot displays the BHL-Europe Portal search results page. At the top, there is a navigation bar with links for Dashboard, Content, Structure, Appearance, People, Modules, Configuration, Reports, and Help. The user is logged in as 'admin'. Below the navigation bar, the page title is 'BHL-Europe Portal'. A search bar is located on the left, and a navigation menu is below it. The main content area shows search results for 'Flora of tropical Africa /By Daniel Oliver ... assisted by other botanists. Volume 1868 v. 1'. A facets sidebar on the right shows 'dc_creator' with values: Oliver, Daniel, (10); Forsskål, Peter, (1); Niebuhr, Carsten, (1). The search results list includes metadata for the first result: creator: Oliver, Daniel; publisher: L. Reeve and co., London; issued: 19th century CE. The second result is 'Flora of tropical Africa /By Daniel Oliver ... assisted by other botanists. Volume 1871 v. 2'.

Figure 4-30: Search results view. Can be customized using XSL transformations.

4.7.4 Metadata View

The metadata view consists of the title of the record and an again customizable stylesheet that transforms the XML-metadata into HTML. Therefore many ways to display the metadata are possible. The Drupal 7 node concept forms that technological basis of the metadata view. We are using the Drupal 7 nodes to store all metadata for a specific item or title. The format is based upon METS and includes lots of different sections. BHL-Europe METS file contains lots of different metadata sections and further tests will show which fields are more relevant and need to be indexed permanently¹⁹. The whole METS document is stored in the nodes' body and can be accessed using stylesheets. Therefore specific metadata views are possible, e.g. more taxonomic oriented display of data, or even a more technological metadata view to view information about scanned pages. We are creating a preset view which is based upon the bibliographic information and thus transforms MODS to HTML. Users are able to create these stylesheets themselves, and choose different stylesheets to display the metadata nodes.

¹⁹ Complex indices need several hours, up to days, to be rebuilt from scratch. Experience shows that corrupted indices need to be rebuilt from scratch. Therefore relevant fields should be selected wisely thus being based on user and expert feedback.



The screenshot shows the BHL-Europe Portal interface. At the top, there is a navigation menu with items like Dashboard, Content, Structure, Appearance, People, Modules, Configuration, Reports, and Help. The user is logged in as 'admin'. The main header is green and displays 'BHL-Europe Portal'. Below the header, there is a search bar and a navigation menu with options like 'Add content', 'Advanced Search 1', 'Advanced Search 2', 'Advanced Search 3', 'Bhle Nodes', 'Browse', and 'Manage saved queries'. The main content area displays the title 'Flora of tropical Africa /By Daniel Oliver ... assisted by other botanists. Volume 1868 v. 1'. Below the title, there are 'View' and 'Edit' buttons. The publication information is 'published by Anonymous (not verified) on Wed, 04/27/2011 - 11:28'. The BHL ID is 'BHLUS:ITEMS/00000000123'. A table of metadata is displayed, with a placeholder for a missing image on the left.

title:	Flora of tropical Africa /By Daniel Oliver ... assisted by other botanists. Volume 1868 v. 1
creator:	Oliver, Daniel,
publisher:	L. Reeve and co., London
identifier:	Abbreviation Fl. Trop. Afr.

Figure 4-31: Metadata view. Can be customized using XSL transformations.

4.7.5 User Profile Personalization

Users can configure settings such as the stylesheets used to display metadata and the metadata fields that are displayed in search masks. The metadata fields displayed in search masks can be selected based upon whatever fields are indexed in the Apache Solr core for advanced search. This enables us to change the search index and make adaptations more easily without having to touch source code.

Users may save search queries (see Figure 4-34). The saved queries can be activated later on, so that the saved query gets executed again. Actually the Solr query string is stored if the user wants to save a certain query.

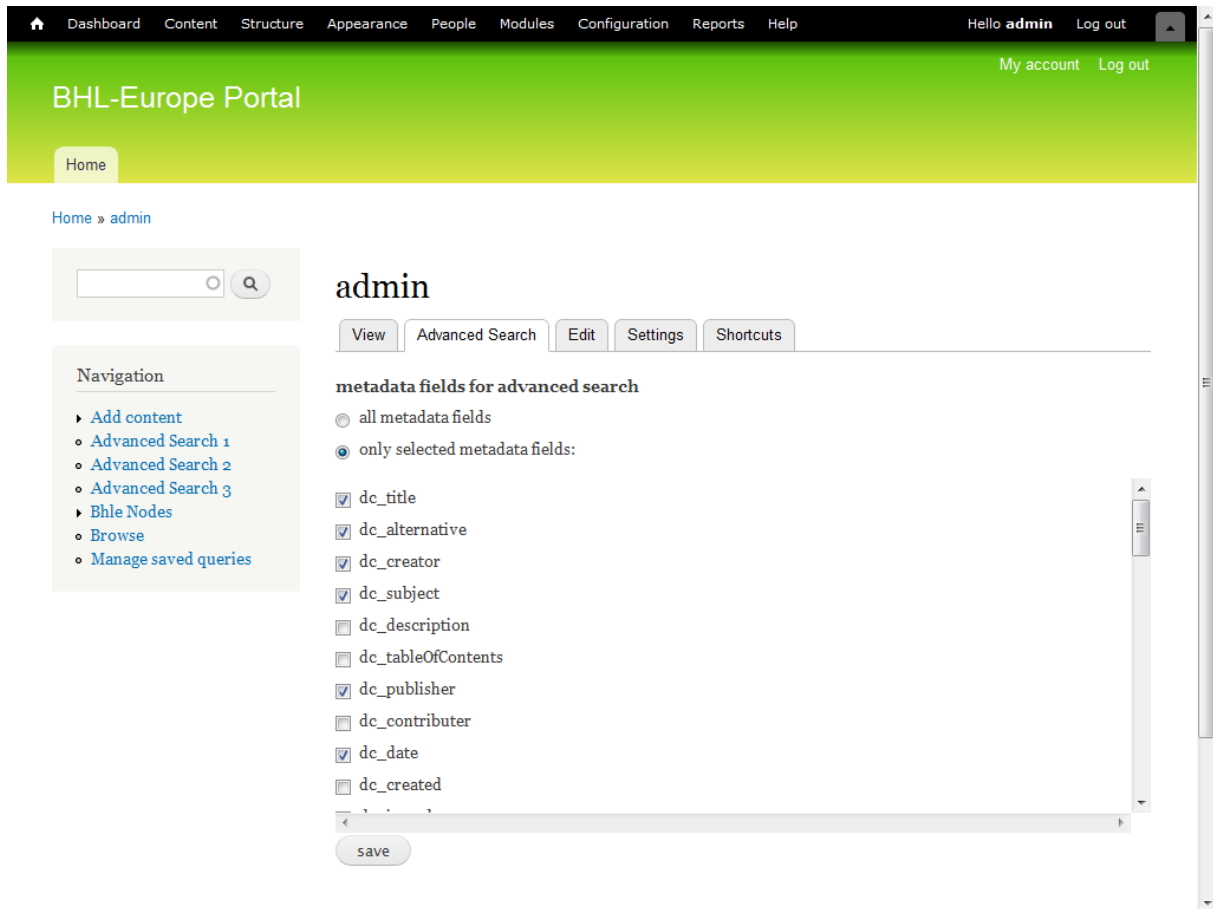


Figure 4-32: User profile personalization. Customizable search fields.

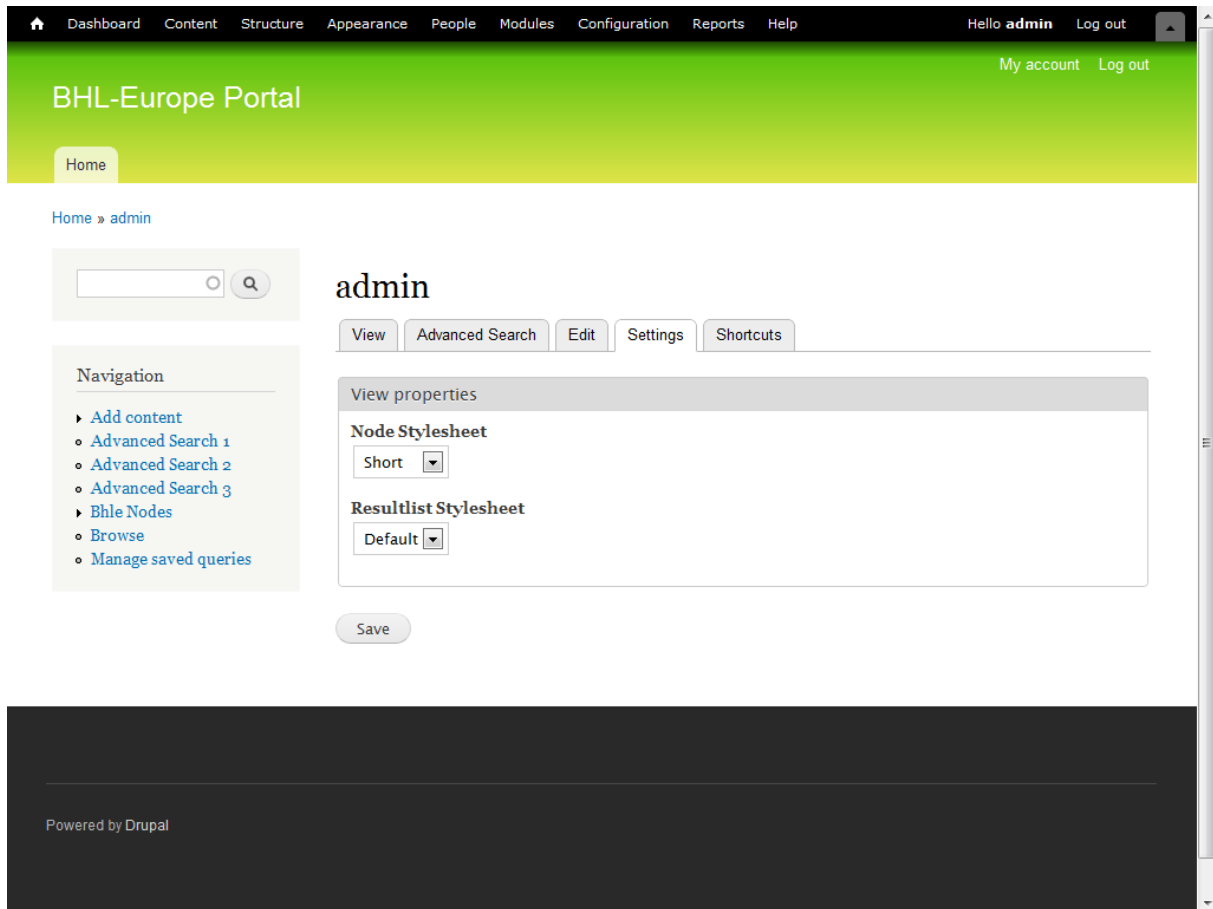


Figure 4-33: User profile personalization. Customizable XSL transformations for views.

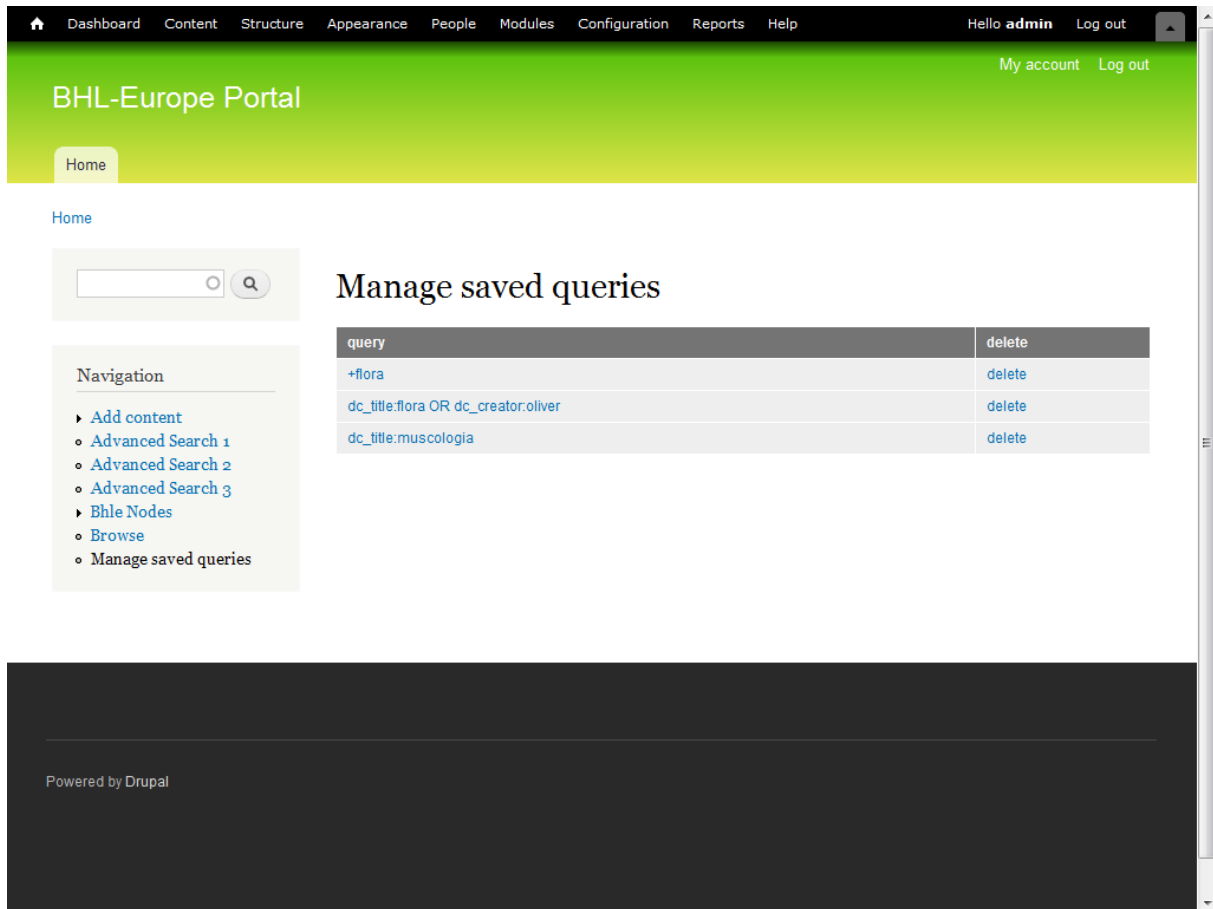
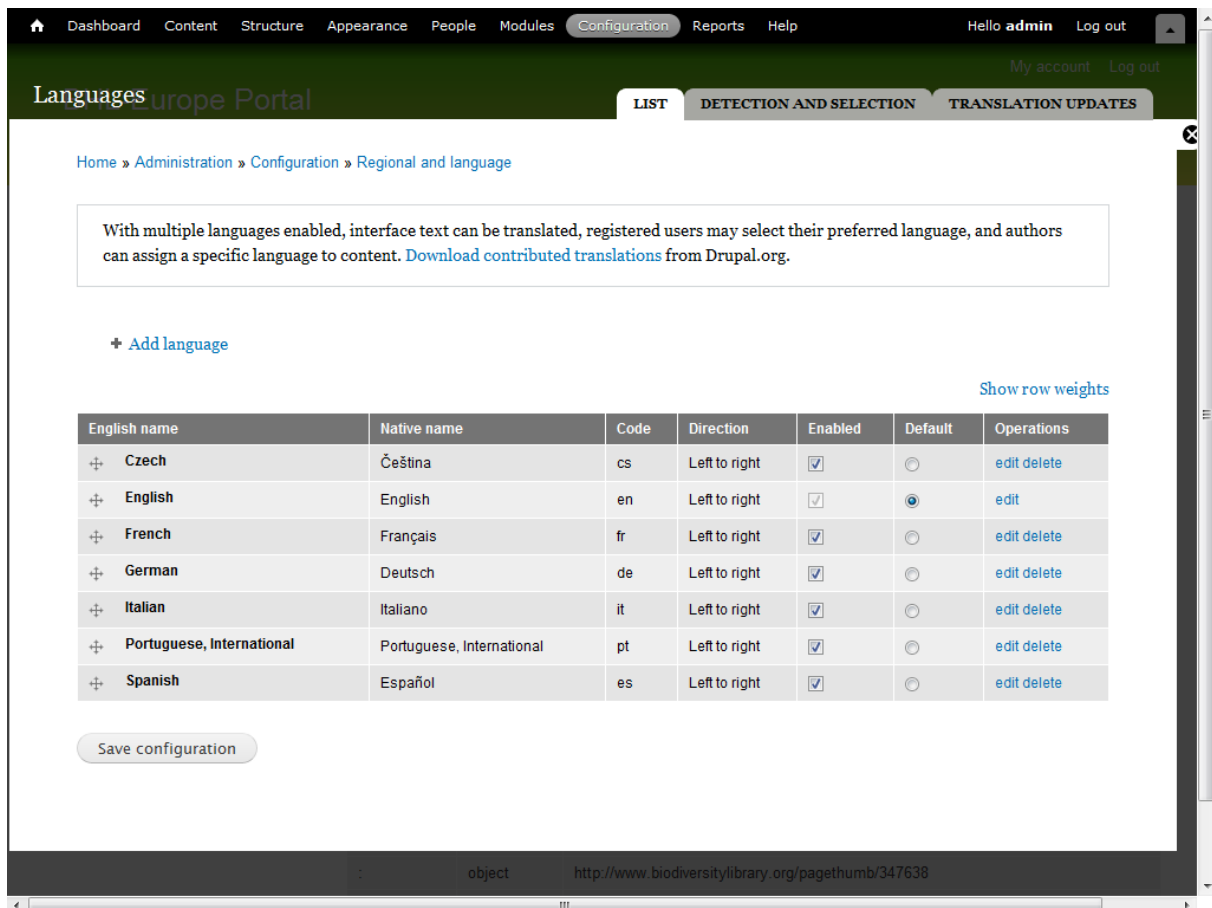


Figure 4-34: Search queries can be saved for later search. The screenshot shows a preliminary view of a user's saved queries.

4.7.6 Multi-lingual UI of the BHL-Europe Portal

Drupal's localization and internationalization modules²⁰ are used to facilitate the translation of the Portal. There are already seven languages implemented: Czech, English, German, French, Italian, Portuguese, Spanish.



With multiple languages enabled, interface text can be translated, registered users may select their preferred language, and authors can assign a specific language to content. [Download contributed translations](#) from Drupal.org.

[+ Add language](#)

[Show row weights](#)

English name	Native name	Code	Direction	Enabled	Default	Operations
+ Czech	Čeština	cs	Left to right	<input checked="" type="checkbox"/>	<input type="radio"/>	edit delete
+ English	English	en	Left to right	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>	edit
+ French	Français	fr	Left to right	<input checked="" type="checkbox"/>	<input type="radio"/>	edit delete
+ German	Deutsch	de	Left to right	<input checked="" type="checkbox"/>	<input type="radio"/>	edit delete
+ Italian	Italiano	it	Left to right	<input checked="" type="checkbox"/>	<input type="radio"/>	edit delete
+ Portuguese, International	Portuguese, International	pt	Left to right	<input checked="" type="checkbox"/>	<input type="radio"/>	edit delete
+ Spanish	Español	es	Left to right	<input checked="" type="checkbox"/>	<input type="radio"/>	edit delete

[Save configuration](#)

Figure 4-35: Multilingual settings showing seven different languages.

²⁰ <http://drupal.org/documentation/modules/locale>
<http://drupal.org/documentation/modules/translation>
http://drupal.org/project/l10n_update
<http://drupal.org/project/potx>

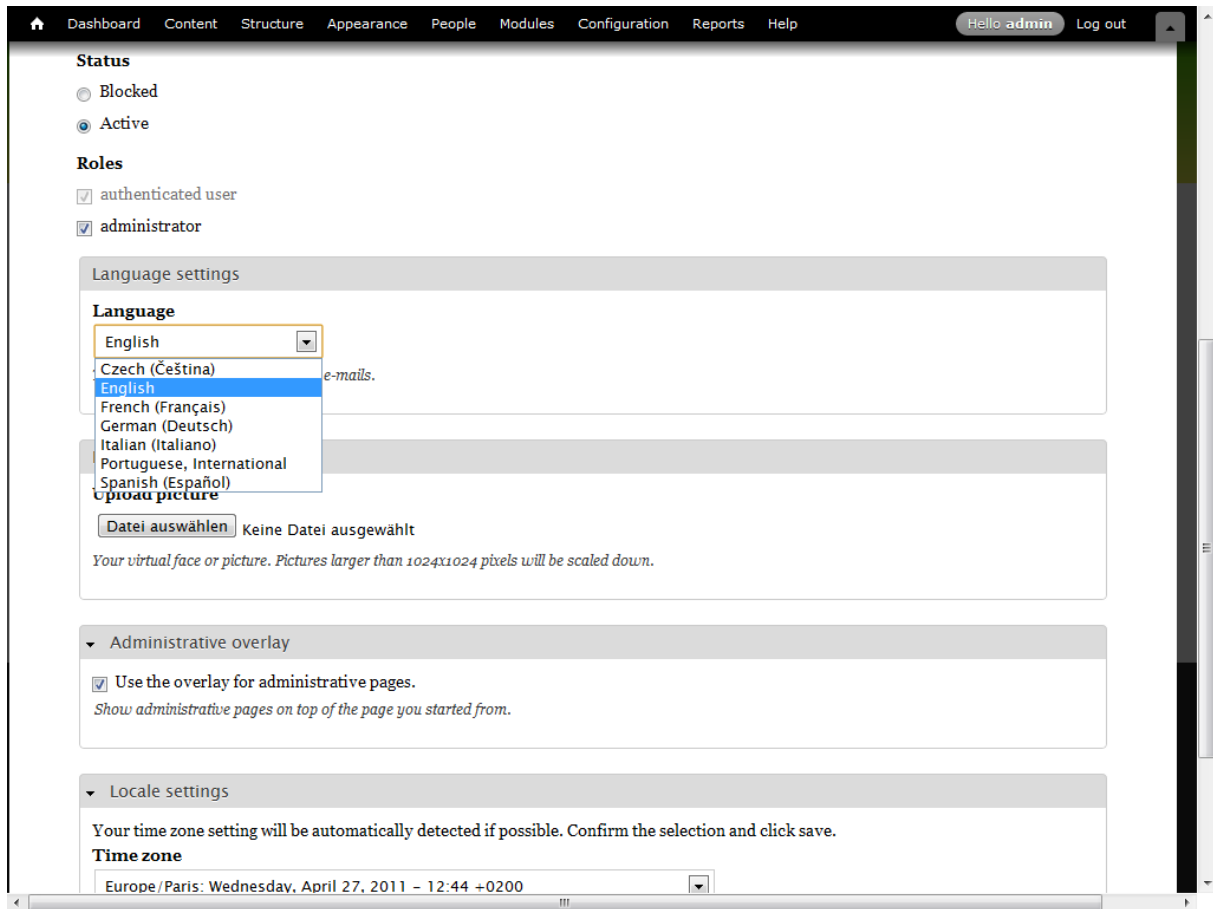


Figure 4-36: Multi-lingual capabilities. Personalization of the portal language in users' profile.

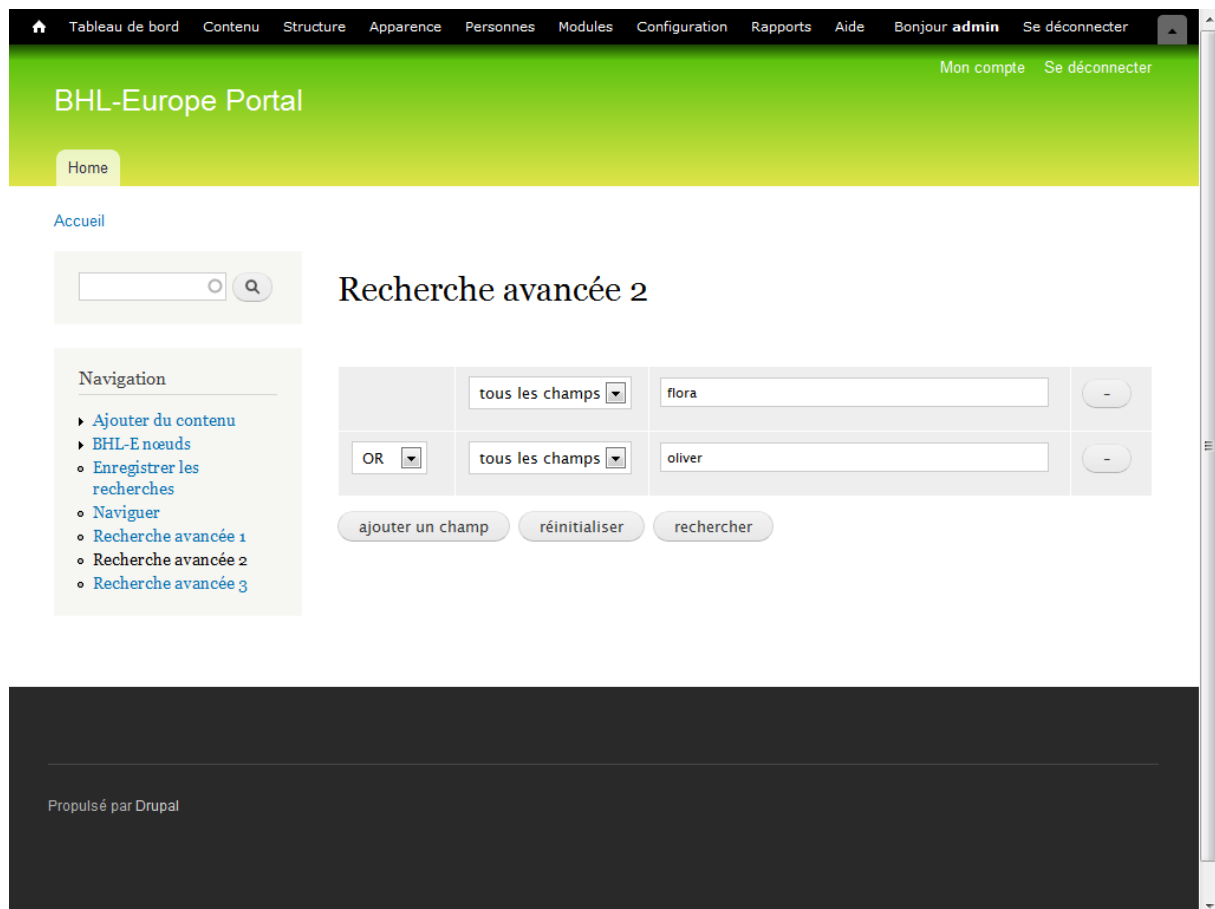


Figure 4-37: Multi-lingual capabilities. Advanced Search 2 localized in French.

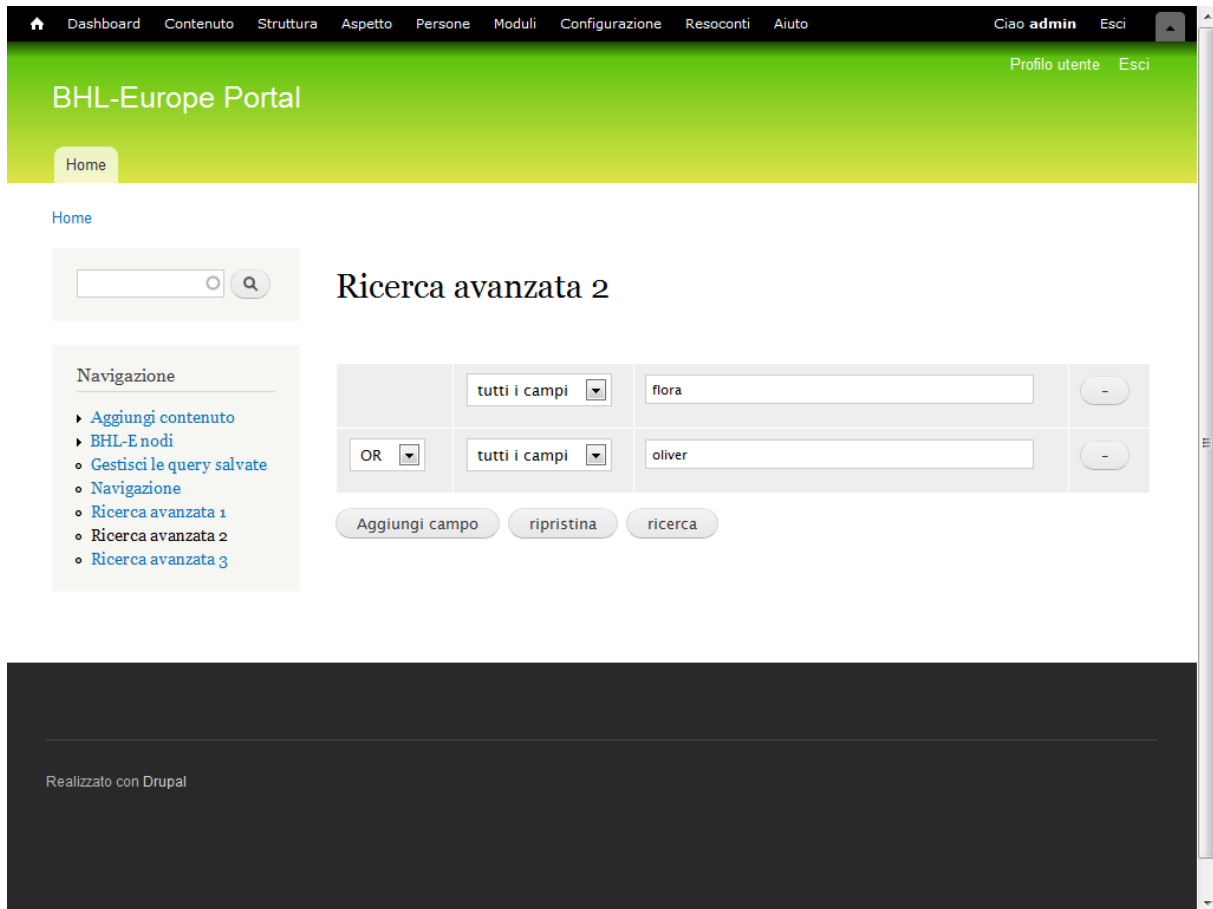


Figure 4-38: Multi-lingual capabilities. Advanced Search 2 localized in Italian.

4.7.7 Content View

The Internet Archive BookReader²¹ is used to implement the content view of each book. Image processing is done on a per request basis. Once an image has been converted it is cached internally and the next request therefore is a lot faster. Preview versions of ingested pages for collections RBGE, RMCA, UH-VIIKKI, CSIC have been created. These providers actually don't have their content online at their websites, so this content is provided for public viewing for the first time. Further development will include full text search inside of the book reader which is based upon OCR and the implementation of a table of contents based upon page types. Also minor changes and bug fixes need to be implemented.

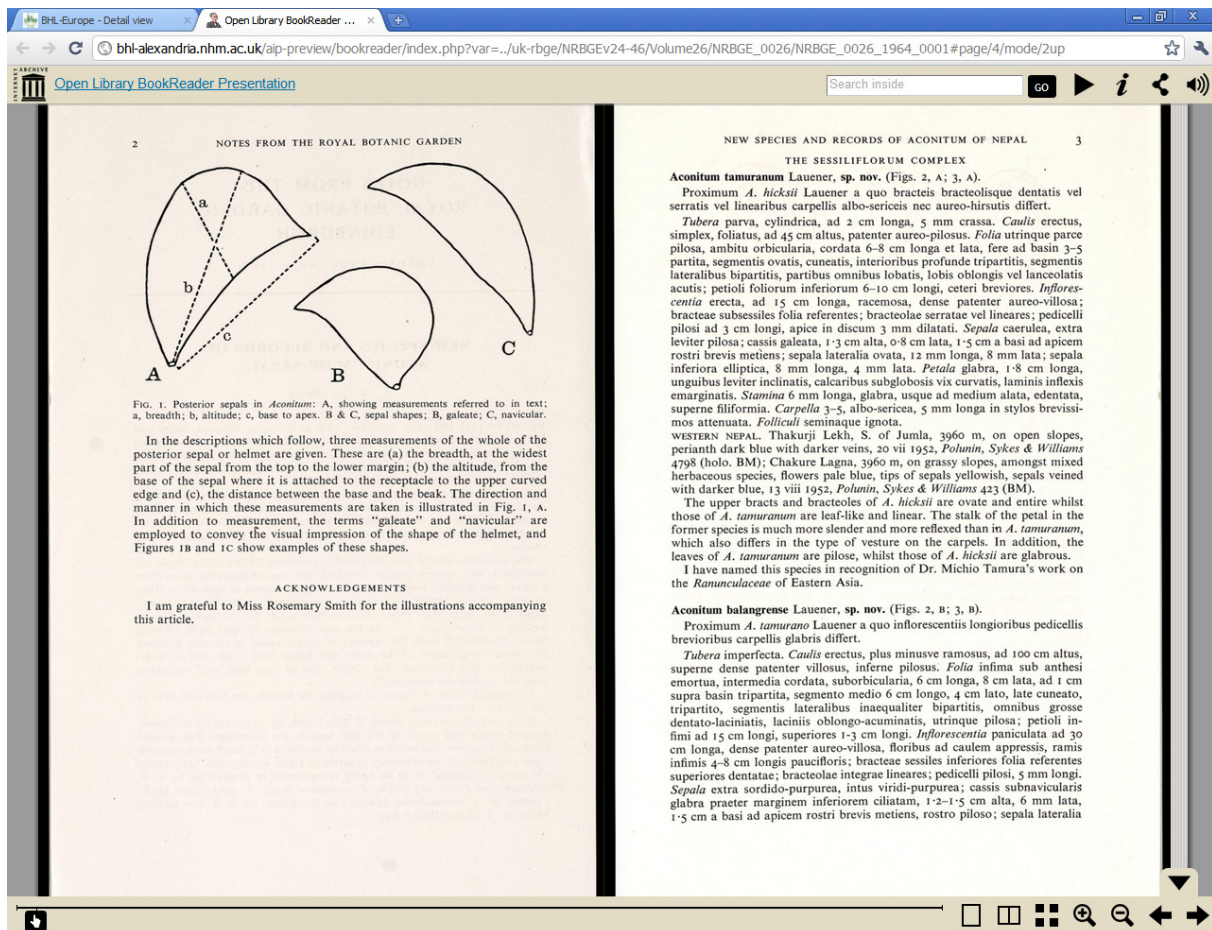


Figure 4-39: Preview of the IA BookReader displaying content²² from collection RBGE.

²¹ <http://openlibrary.org/dev/docs/bookreader>

²² http://bhle-dev-1.nhm.ac.uk/aip-preview/bookreader/index.php?var=../uk-rbge/NRBGEv24-46/Volume26/NRBGE_0026/NRBGE_0026_1964_0001#page/4/mode/2up

4.7.8 Browse Index

Browsing is implemented primarily to enable users to discover content and metadata. In future versions facets are going to be used here to give a better overview over the contents of BHL-Europe portal.

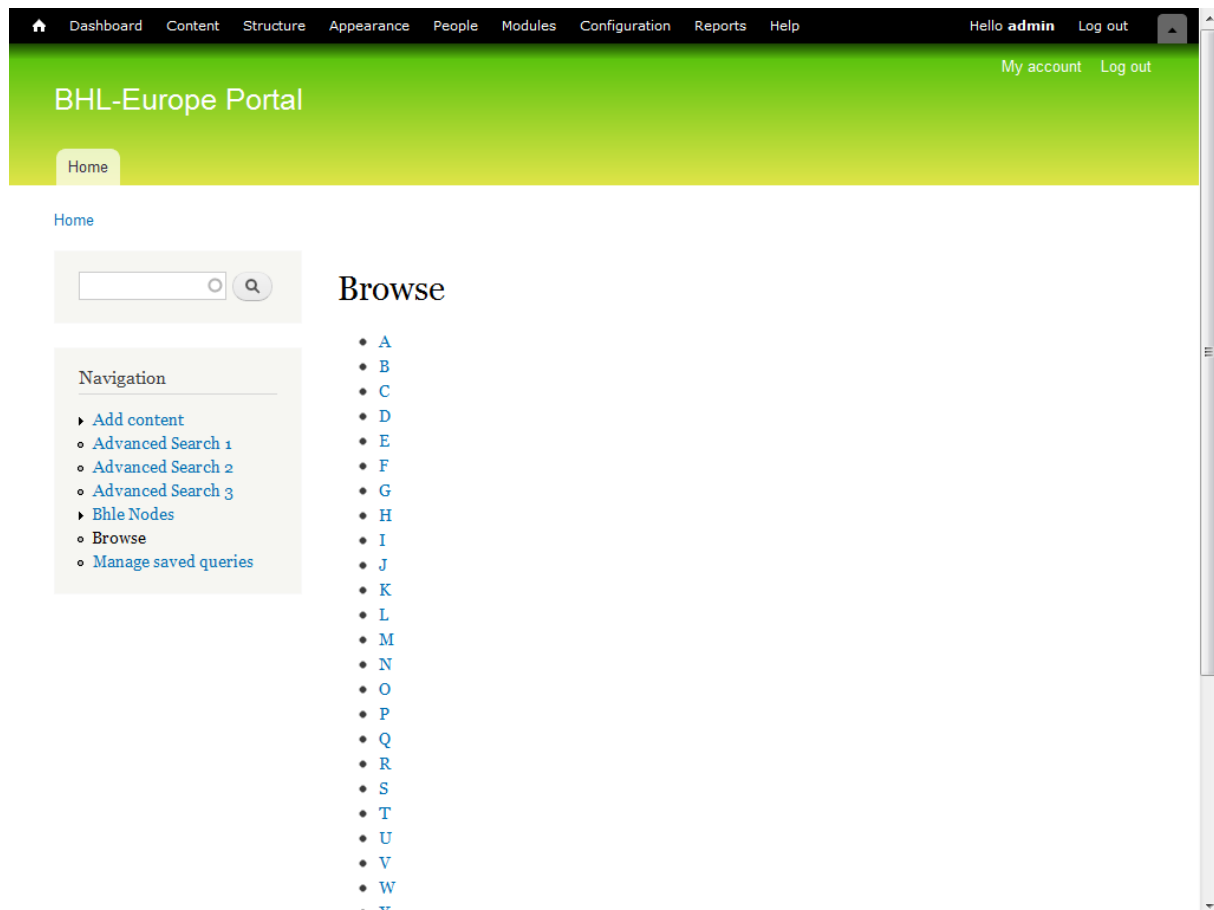


Figure 4-40: Browse Index is implemented using an A-Z map.

4.7.9 Drupal 7

Drupal 7 is a content management system based on PHP5+. It offers various core-functionalities for user-management, page-layout, site administration etc. It has a series of concepts which are key for the BHL-Europe portal: Drupal 7 modules are developed by migrating legacy code from the German prototype into a Drupal 7 API conformant code base. This makes all BHL-Europe Portal features very maintainable and guarantees sustainability. It is one of the goals to base development on systems which enable the community to further sustain and develop it once the project is finished.

4.7.9.1 Modules²³

Additional functionality can be integrated via modules. The BHL-Europe Portal component is in fact a collection of modules. This provides the ideal ground for porting functionality from the German prototype to the BHL-Europe portal. There are more than 1500 Drupal 7 modules registered out of more than 7800 in total (versions including 7 and below).

4.7.9.2 Nodes

A node is a Drupal object that stores content. These nodes can be articles, blog entries, etc.. In the case of BHL-Europe we create our own nodes. Nodes store metadata and can be seen as a cache which is updated through the Access component. A 'Metadata node' has been developed which holds all METS content and displays it using a default stylesheet. Users can customize this stylesheet or add their own.

4.7.9.3 Fields²⁴

Fields extend nodes and are new in Drupal 7. Each node has an attached body-field, that stores the content of the node. In the case of BHL-Europe its content is the XML-METS metadata of the record. But a node can also have additional fields attached to it. The BHL-Europe-Id of the record is extracted from METS and saved in a field. This unique identifier is used throughout all of BHL-Europe and also used to update the Drupal 7 cache using the Access component.

4.7.9.4 Drupal API²⁵

The Drupal API (Application Programming Interface) is divided into several parts, of which are especially used:

- Module system: Modules can implement a set of hooks, which are functions with predefined names, that are called from the Drupal core in order to execute the functionality of the module.
- Menu system: Menu entries on the page are generated here. Also paths components such as search masks, result sets, configuration pages, etc. are registered.
- Form generation: The input fields for search, configuration, etc. are generated via the forms API.
- Theme system: Components of themes can be accessed in the source code via the theme API.

4.7.9.5 Solr integration into Drupal 7

Solr is an indexing software which is used to index all BHL-Europe-nodes. A search in the portal uses the index in order to find the desired nodes.

²³ <http://drupal.org/project/Modules>

²⁴ <http://drupal.org/community-initiatives/drupal-core/fields>

²⁵ <http://api.drupal.org/api/drupal>

Indexed is the descriptive metadata section of the XML-METS from each node. For the advanced search, additional metadata-fields are index separately. So searches can be executed in one specific metadata-field.

4.7.9.6 XML Stylesheets

Because the metadata for each record is saved in XML, XSL is used to generate HTML from the metadata. This is an easy and interchangeable way to provide a visual appealing view of the metadata. Therefore XSL is used for every view module.

4.8 GUID Mint/Resolver

Following on from discussions held during the meeting in Alexandria; we have reviewed for suitability with BHL-Europe's architecture the current GUID approach implemented by Bibliotheca Alexandrina, who use Handle.net as a resolution service. (see Appendix III.III)

We have found similarities in purpose and after feedback from this meeting intend to implement the service to meet BHL-Europe needs via two components:

- a GUID Mint: opaque GUIDs will be generated on request and associated with objects during Scan Management. Where the Content Provider scan process has already been achieved without GUIDs, the Pre-Ingest process will carry out this association.
- a Handle resolver: implemented at regional level, enabling growth to a global architecture. Handles linking the GUID to the portal landing page URL for each object will be created at first opportunity during the ingest to Fedora. The Handle service will offer resolution for User requests from DOI to the object.

5 The ingestion process from BHL-Europe to Europeana

Once the BHL-Europe content partners have uploaded their metadata and image files into the FTP directories of the London Natural History Museum's server the ingestions process to BHL-Europe and also to Europeana starts.

The content providers acknowledge their finalization of upload based on the Pre-Ingest guidelines to the WP2 lead. Next a schema mapping tool developed by WP3 converts all metadata supplied in the MARC format (MACHINE-Readable Cataloging, <http://www.loc.gov/marc/>) into the BHL-Europe METS (Metadata Encoding and Transmission Standard, <http://www.loc.gov/standards/mets>) format. The descriptive metadata of the BHL-Europe METS file is transformed into MODS (Metadata Object Description standard, <http://www.loc.gov/standards/mods>).

Once these mapped files are available on the NHM FTP server the ingest process to Europeana is initiated.

In this phase of the project whilst the final BHL-Europe portal is still under development the established OAI-provider services of the BHL-Europe prototype are used for the direct provision of data to Europeana.

The METS files are mapped to the Europeana ESE standard (www.europeana.eu) which is the current data standard for Europeana and which is also integral part of the new Europeana Data Model (EDM). For the mapping process the well-established open source business intelligence suite Pentaho (<http://www.pentaho.com/>) is applied.

The toolset transforms the METS files into valid ESE files which may look like this:

```
<europeana:record xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms/"
xsi:schemaLocation="http://www.europeana.eu/schemas/ese/
http://www.europeana.eu/schemas/ese/ESE-V3.3.xsd">
  <dc:title>Notes from the Royal Botanic Garden Edinburgh: Volume 6, addition to numbers
  XXIX-XXX (January 1917) (part name)</dc:title>
  <dc:publisher>HMSO (Edinburgh)</dc:publisher>
  <dc:type>Text</dc:type>
  <dc:format>print</dc:format>
  <dc:identifier>d2bd6a0e-d46a-4f0c-9486-c6060235b201</dc:identifier>
  <dc:identifier>0080-4274</dc:identifier>
  <dc:identifier>26774</dc:identifier>
  <dc:language>eng</dc:language>
  <dc:relation>Edinburgh Journal of Botany</dc:relation>
  <dcterms:tableOfContents>Plates I-XXXVII illustrating Mr Takeda's paper on the Genus
  Mahonia.</dcterms:tableOfContents>
  <dcterms:issued>1921</dcterms:issued>
  <europeana:object>http://bhle-dev-1.nhm.ac.uk/aip-preview/uk-rbge/NRBGEv1-
  23/Volume06/NRBGE_0006/NRBGE_0006_1917_0029-30_PLATES/thumbnail.jpg</europeana:object>
  <europeana:provider>BHL-Europe </europeana:provider>
  <europeana:type>TEXT</europeana:type>
  <europeana:rights>http://www.europeana.eu/rights/rr-f/</europeana:rights>
  <europeana:dataProvider>Royal Botanic Garden Edinburgh; United
  Kingdom</europeana:dataProvider>
  <europeana:isShownBy>http://bhle-dev-1.nhm.ac.uk/aip-preview/uk-rbge/NRBGEv1-
  23/Volume06/NRBGE_0006/NRBGE_0006_1917_0029-30_PLATES</europeana:isShownBy>
  <europeana:isShownAt>http://bhl.ait.co.at/index.php?form=display&amp;oiid=RBGE/D2BD6A0ED46A4F
  0C9486C6060235B201</europeana:isShownAt>
  </europeana:record>
```

Figure 5-1: Europeana ESE data

5.1 The Europeana fields

<europeana:isShownBy>

This field must contain the link to the digital object.

At this stage of the project a first book viewer version was established at the NHM server using the Open Library BookReader developed by InternetArchive. The link to this online viewer is added in the respective Europeana field.

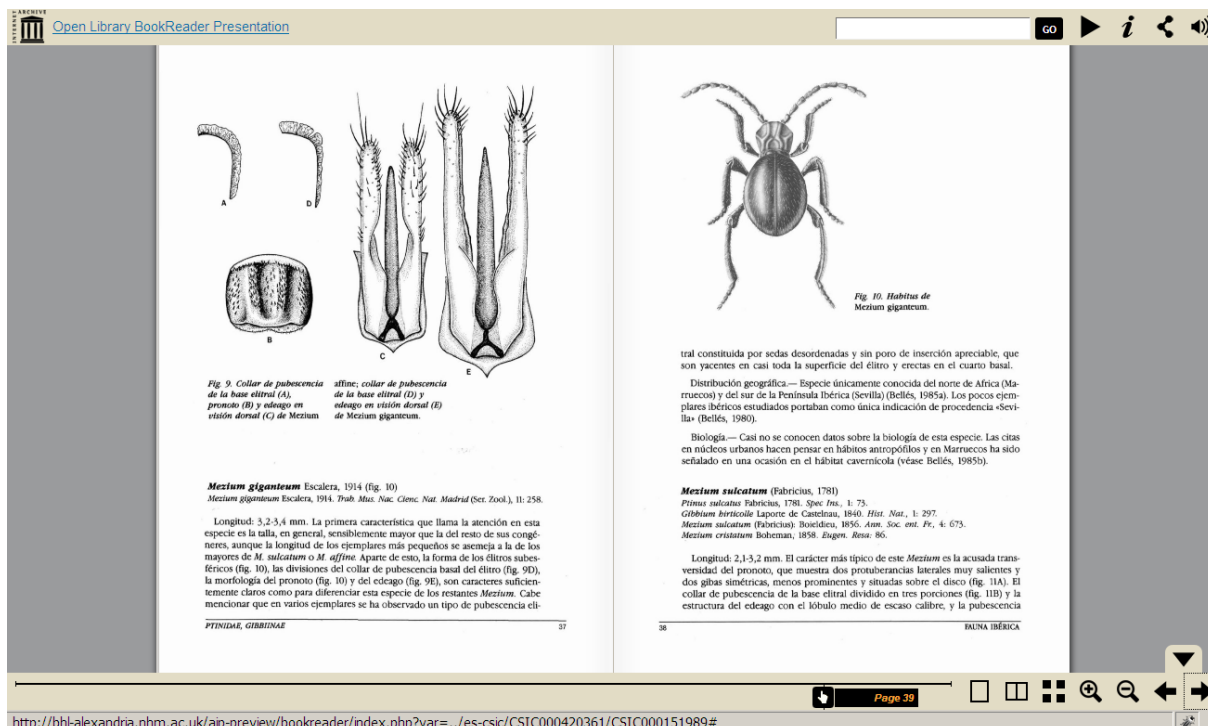


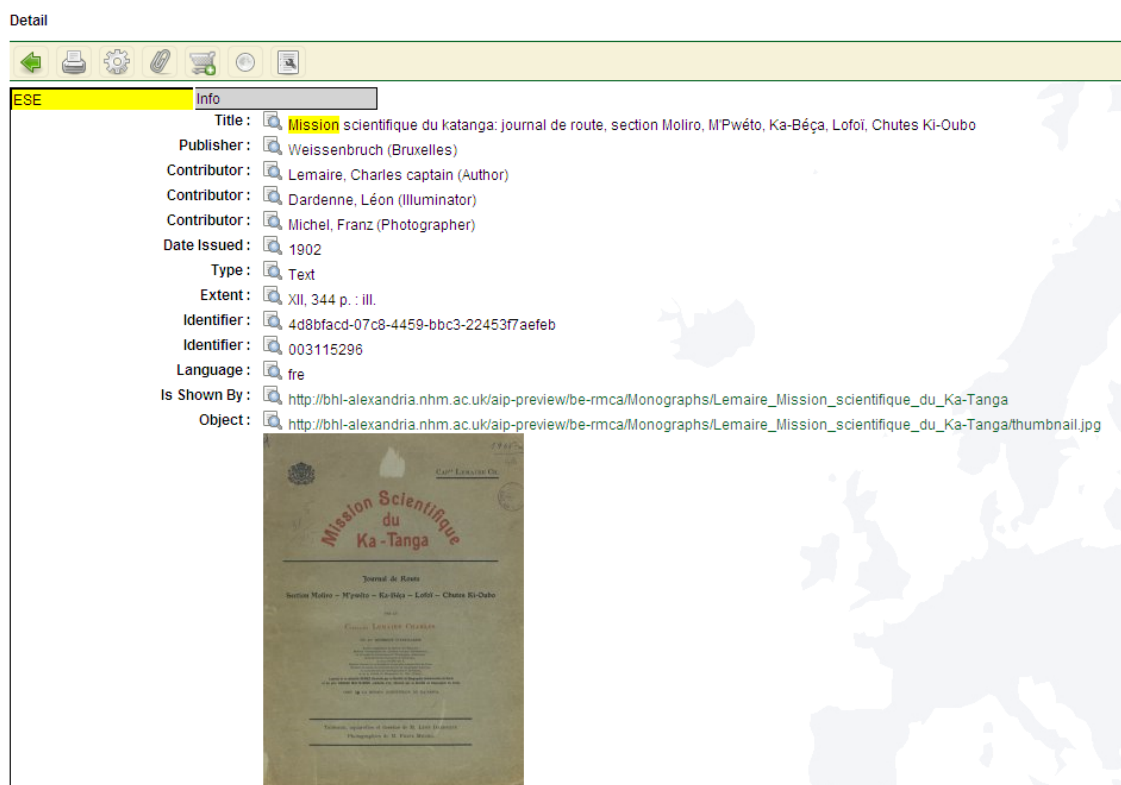
Figure 5-2: BookReader View of BHL-Europe content

<europeana:isShownAt>


This field will in future link directly to Website of the final BHL-Europe portal where the book is described.


The portal is still under development, but this fact was no obstacle to the Europeana ingests as in the very first ingestion phase partners with own online representations of their content (BHL-US, NAT, LANDOE, UBER, UBBI, UCPH) were selected. And for the partners of the second major ingestion phase (RMCA, RBGE, UHVIKKI, CSIC) the display of the BHL-Europe Prototype where the OAI provider for Europeana is located was used.


Detail





ESE Info


Title:  [Mission scientifique du katanga: journal de route, section Moliro, MPwéto, Ka-Béça, Lofoi, Chutes Ki-Oubo](#)


Publisher:  Weissenbruch (Bruxelles)


Contributor:  Lemaire, Charles captain (Author)


Contributor:  Dardenne, Léon (Illustrator)


Contributor:  Michel, Franz (Photographer)


Date Issued:  1902


Type:  Text


Extent:  XII, 344 p. : ill.

Identifier:  4d8bfacd-07c8-4459-bbc3-224537aefeb

Identifier:  003115296

Language:  fre

Is Shown By:  http://bhl-alexandria.nhm.ac.uk/aip-preview/be-rmca/Monographs/Lemaire_Mission_scientifique_du_Ka-Tanga

Object:  http://bhl-alexandria.nhm.ac.uk/aip-preview/be-rmca/Monographs/Lemaire_Mission_scientifique_du_Ka-Tanga/thumbnail.jpg

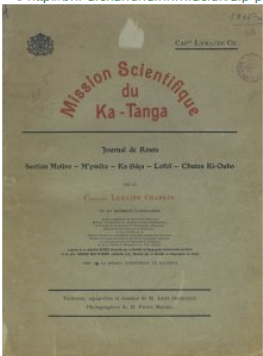


Figure 5-3: Detail view in the BHL-Europe prototype

<europeana:dataProvider>

The value of this field displays the official name of the content delivering institution and is depicted in the Europeana result list display above the name of the Europeana aggregator. The institutions acknowledged to WP3 which version of their name they want to use for Europeana. The value of this field is automatically inserted during the ingest to the Europeana OAI provider.

<europeana:provider>

The value of this field shows the name of the institution, project or aggregator that directly supplies data to Europeana. The value “BHL-Europe” is added automatically.

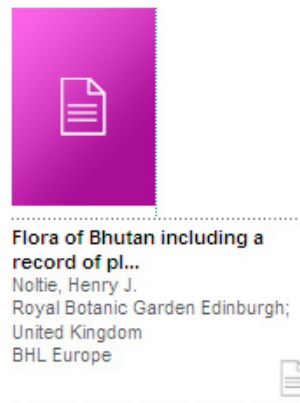
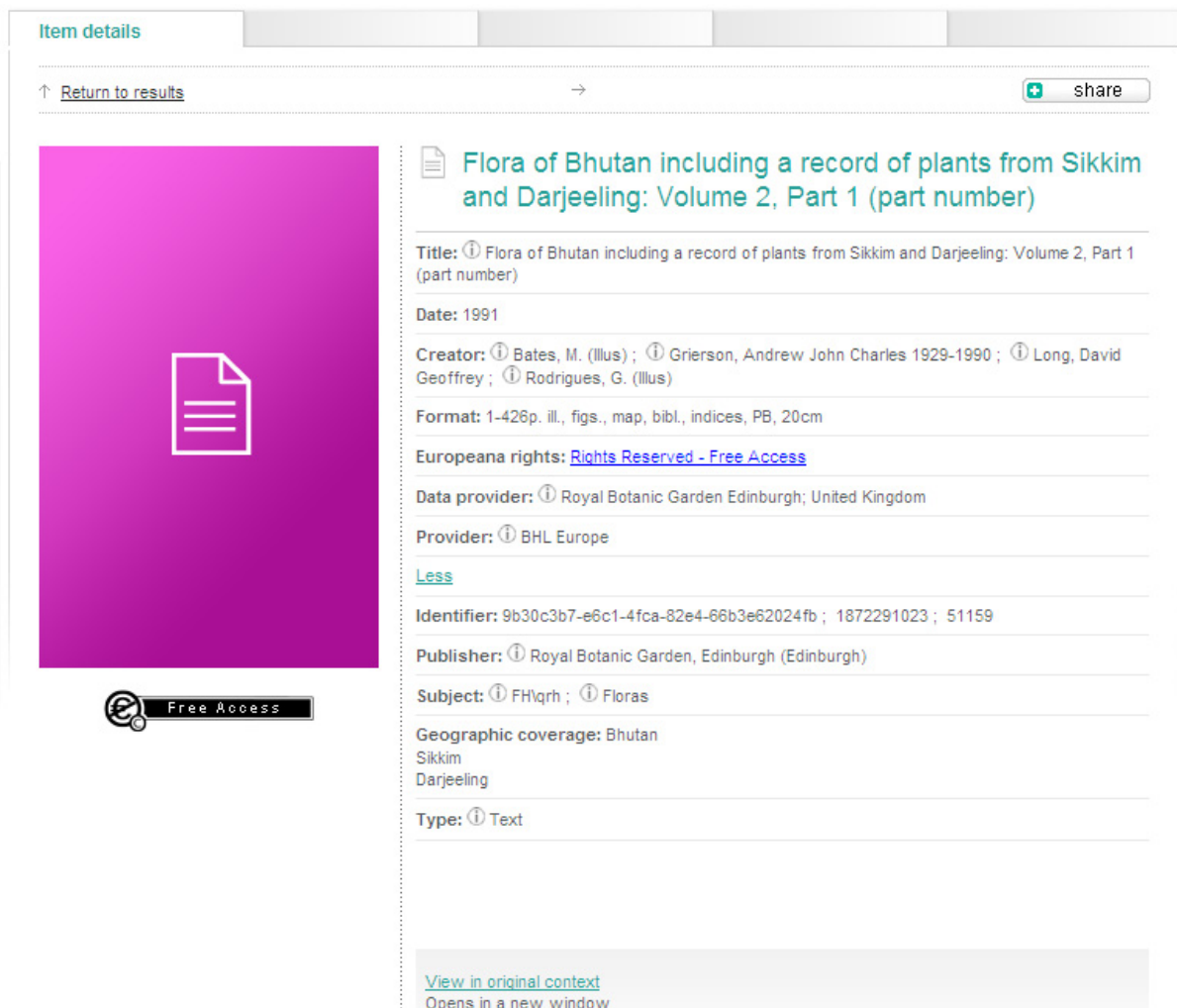


Figure 5-4: Display of the europeana:dataProvider field and the europeana:provider field


<europeana:rights>


With the release of the ESE standard version 3.3 (autumn 2010) the data field europeana:rights was introduced. This field includes a statement about the rights status of the digital objects described in the metadata submitted to Europeana. The value of the field must be an URL linking to a license statement. WP3 provided to the content providers samples of statements and they finally chose what license URL they what to have inserted in this field. A license was added to the content automatically if it applied to all content/records of the partner. If the partner submitted records with different license URLs the information had to be within the metadata.







Item details

[Return to results](#) share

 **Flora of Bhutan including a record of plants from Sikkim and Darjeeling: Volume 2, Part 1 (part number)**


Title:  Flora of Bhutan including a record of plants from Sikkim and Darjeeling: Volume 2, Part 1 (part number)


Date: 1991

Creator:  Bates, M. (Illus) ;  Grierson, Andrew John Charles 1929-1990 ;  Long, David Geoffrey ;  Rodrigues, G. (Illus)

Format: 1-426p. ill., figs., map, bibl., indices, PB, 20cm


Europeana rights: [Rights Reserved - Free Access](#)



Data provider:  Royal Botanic Garden Edinburgh; United Kingdom

Provider:  BHL Europe


[Less](#)

Identifier: 9b30c3b7-e6c1-4fca-82e4-66b3e62024fb ; 1872291023 ; 51159

Publisher:  Royal Botanic Garden, Edinburgh (Edinburgh)

Subject:  FHVqrh ;  Floras

Geographic coverage: Bhutan
Sikkim
Darjeeling

Type:  Text

[View in original context](#)
Opens in a new window

Figure 5-5: Display of the europeana:rights field together with license icon

<europeana:object>

This field provides for Europeana the link to the thumbnail for the record. It was suggested during a project meeting to use the image qualified as “title” for that purpose. Whenever there was no image with “title” in the filename the image with “cover” in the filename was used as thumbnail image.

During the Pre-Ingest process the thumbnails for the records are generated.

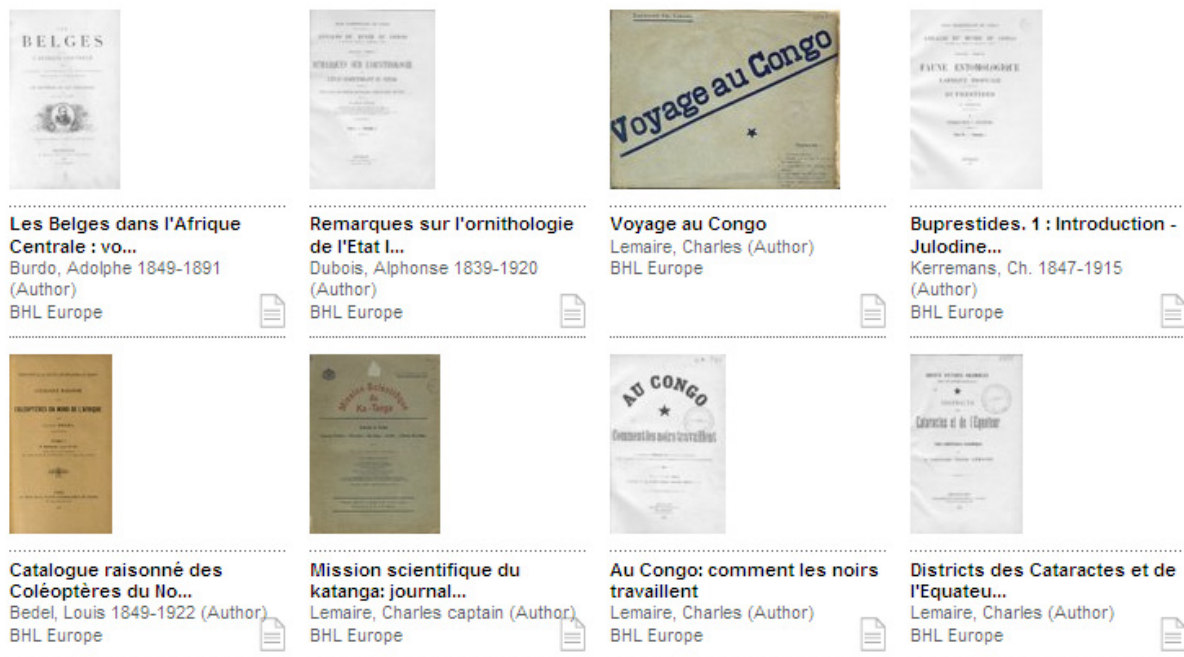


Figure 5-6: Thumbnails for Europeana

<europeana:type>

The value of this field must be one of: TEXT, IMAGE, VIDEO, SOUND. For BHL-Europe content the value “Text” is added automatically to each record as all of the BHL-Europe content is library content/books. We do not ingest separate images out of books into Europeana but always whole books or whole articles.

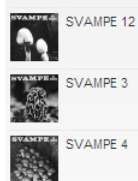
5.2 Europeana content checking

Once the content of the partners is in the OAI repository the partners receive a link to their content in the Europeana content checker tool where they can preview their records. In general about 100 test items per partner are uploaded to the content checker as the tool is not suited for mass uploads but rather for a “look-and-feel” preview checking.



Related content:

Items


[See all related items](#)

Actions:

[Add a tag](#) →

[Share with a friend](#) →


[Save to My Europeana](#) →

[Login](#) | [Register](#)
[My Europeana](#) [Communities](#) [Partners](#) [Timeline \(beta\)](#) [Thought lab](#) [Choose a language](#)

[Advanced search](#)

Matches for: europeana_collectionName: 08703*

[Return to results](#) ← →



SVAMPE 2

Title: SVAMPE 2

Date: 2010-01-09; 2011-04-10

Creator: Danish Mycological Society; Foreningens til Svampekundskabens Fremme ; Natural History Museum, Copenhagen; Statens Naturhistoriske Museum

Description: The journal of the Danish Mycological Society (Forening til Svampekundskabens Fremme), published bi-annually. The journal started in 1980, taking over from the journal FRIESIA. It contains articles on fungi and other mycological issues connected to mycology in Denmark. Online table of contents to authors, articles and photos can be found at the society homepage, www.svampe.dk.

Format: 400 dpi; 52 pages; PDF

Source: pdf of the printed version

Rights: <http://www.europeana.eu/rights/rr-f/>

Data Provider: University of Copenhagen; Denmark

Provider: BHL Europe

Europeana Rights: <http://www.europeana.eu/rights/rr-f/>

[Less](#)

Identifier: <http://www.svampe.dk/svampe-2/>; ISSN 0106-7451

Publisher: Danish Mycological Society; Foreningens til Svampekundskabens Fremme

Subject: 20th century CE/1980s; Denmark; Europe/Northern Europe/Denmark; Fungi; Mycology

Figure 5-7: Detail view in the content checker tool

Partners could give WP3 feedback on their content presentation in the content checker tool. Their feedback was incorporated before the data was finally provided to Europeana.

At this time Europeana office also receives the link to the new content in the content checker and provides feedback.

On April 21st 2011 a new Europeana harvest of the following BHL-Europe sets was requested:

- * BHLUS (87572)
- * CSIC (9)
- * LANDOE (3511)
- * NBN (3516)
- * RBGE (183)
- * RMCA (52)
- * UBBI (1634)
- * UBER (60)
- * UCPH (28)
- * UHVIKKI (30)

Europeana started the harvest on April 22nd in the morning. The data will be part of the new Europeana publication in early May 2011.

Figure 5-8: Books for Europeana (UBER)

Figure 5-9: Articles for Europeana (UBBI)

I Acronyms

AIP	Archival Information Package
API	Application Programming Interface
AS	Archival Storage
AUC	Archivist Use Case
BHL	Biodiversity Heritage Library
BHL-US	Smithsonian Institution, Natural History Museum London, US, UK
CRC	Cyclical Redundancy Check
CSIC	Consejo Superior de Investigaciones Cientificas, ES
DI	Descriptive Information
DIP	Dissemination Information Package
DM	Data Management
ESE	Europeana Semantic Elements
ETL	Extract, Transform, Load
Europeana	European Digital Library
GRIB	Global References Index to Biodiversity
GUID	Global Unique Identifier
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
ING	Ingest
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LANDOE	Land Oberösterreich, AT
LOCKSS	LOCKSS (Lots of Copies Keep Stuff Safe), based at Stanford University Libraries, is an international community initiative that provides libraries with digital preservation tools and support so that they can easily and inexpensively collect and preserve their own copies of authorized e-content
MARC	MACHINE-Readable Cataloging
METS	Metadata Encoding and Transmission Standard
MIME type	Internet Media Type
MODS	Metadata Object Description standard
NAT	Stichting Nationaal Natuurhistorisch Museum, NL

OAI	Open Archives Initiative
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
OAIS	Open Archival Information System
OCR	Optical Character Recognition
ODBC	Open Database Connectivity
OLEF	Open Literature Exchange Format
PDI	Preservation Description Information
PI	Pre-Ingest
PUC	Producer Use Case
RBGE	Royal Botanic Garden Edinburgh, UK
RDF	Resource Description Framework
REST	Representational state transfer
RMCA	Royal Museum for Central Africa, BE
RSS	Really Simple Syndication (RSS) is a lightweight XML format designed for sharing headlines and other Web content.
SIP	Submission Information Package
SOAP	Simple Object Access Protocol providing a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralised, distributed environment using XML proposed under the W3C.
SWORD	Simple Webservice Offering Repository Deposit
UBBI	University of Bielefeld, DE
UBER	Humboldt-Universität zu Berlin, DE
UCPH	University of Copenhagen, DK
UHVIKKI	Helsingin yliopisto, FI
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	EXtensible Markup Language
XSLT	XSLT (XSL Transformations) is a declarative, XML-based language used for the transformation of XML documents into other XML documents.

II Figures

Figure 2-1: SIP Preparation Workflow	7
Figure 3-1: BHL-Europe Architecture Diagram	9
Figure 4-1: Metadata Mapping to OLEF.....	11
Figure 4-2: Schematic outline of OLEF structure.....	12
Figure 4-3: Pre-Ingest process as documented in Deliverable 3.4.....	13
Figure 4-4: Comparison of Merritt vs. Archivematica micro-services for Pre-Ingest.	14
Figure 4-5: Pre-Ingest Workflow showing pool with swim lanes / part 1.....	15
Figure 4-6: Pre-Ingest Workflow showing pool with swimlanes / part 2.....	16
Figure 4-7: OAIS function model showing the location of Pre-Ingest.	17
Figure 4-8: OAIS functions of ingest showing which areas are also partly covered by the Pre-Ingest component.	17
Figure 4-9: Web application for the Pre-Ingest Tool. Submitted directories can be selected for further processing.....	18
Figure 4-10: Visualization of different messaging channels and service activators (such as loading-ramp, security-check, boxing, create-aip, ...) of Pre-Ingest Tool. Starting from loading gateway in the upper left until create-aip in the lower right.	20
Figure 4-11: Deployment diagram of the Pre-Ingest tool.	21
Figure 4-12: Pentaho Kettle transformation from Excel to XML using multiple enrichment steps in between.	23
Figure 4-13: Deployment diagram of Ingest component showing the script used for batch ingest of AIPs and the web application Directory Ingest Service (in the context of Fedora Commons).	24
Figure 4-14: The ingest component covers the process required to ingest AIPs into the Archival storage system.	24
Figure 4-15: Ingest architecture as documented in Deliverable 3.4.....	25
Figure 4-16: Archival Storage Architecture.....	26
Figure 4-17: Archival Storage Deployment Diagram.....	28
Figure 4-18: Data Management Architecture.....	29
Figure 4-19: Data Management Deployment Diagram.....	30

Figure 4-20: Deployment diagram of Access component.....	31
Figure 4-21: Access processes covered in the context of BHL-Europe.....	31
Figure 4-22: Access Architecture as planned in D3.4.....	32
Figure 4-23: Deployment diagram showing Drupal 7 modules as components (core features and metadata node). Ontology services are used for taxonomic enrichment.....	33
Figure 4-24: Simple search screenshot from current development version.	34
Figure 4-25: Advanced search from current development version. Multiple fields can be select.....	35
Figure 4-26: Advanced search example as seen on Europeana.	35
Figure 4-27: Advanced search (Google-like) from current development version.	36
Figure 4-28: Advanced search as seen on www.google.com.....	36
Figure 4-29: Advanced search screenshot from current development version. Search syntax can be manually entered and thus provides an optimised interface for experts.....	37
Figure 4-30: Search results view. Can be customized using XSL transformations.	38
Figure 4-31: Metadata view. Can be customized using XSL transformations.....	39
Figure 4-32: User profile personalization. Customizable search fields.	40
Figure 4-33: User profile personalization. Customizable XSL transformations for views.....	41
Figure 4-34: Search queries can be saved for later search. The screenshot shows a preliminary view of a user's saved queries.....	42
Figure 4-35: Multilingual settings showing seven different languages.	43
Figure 4-36: Multi-lingual capabilities. Personalization of the portal language in users' profile.	44
Figure 4-37: Multi-lingual capabilities. Advanced Search 2 localized in French.....	45
Figure 4-38: Multi-lingual capabilities. Advanced Search 2 localized in Italian.....	46
Figure 4-39: Preview of the IA BookReader displaying content from collection RBGE.....	47
Figure 4-40: Browse Index is implemented using an A-Z map.	48
Figure 5-1: Europeana ESE data	51
Figure 5-2: BookReader View of BHL-Europe content	52
Figure 5-3: Detail view in the BHL-Europe prototype	53
Figure 5-4: Display of the europeana:dataProvider field and the europeana:provider field ...	54
Figure 5-5: Display of the europeana:rights field together with license icon	55



Figure 5-6: Thumbnails for Europeana	56
Figure 5-7: Detail view in the content checker tool	57
Figure 5-8: Books for Europeana (UBER).....	58
Figure 5-9: Articles for Europeana (UBBI)	58

III Appendices

III.I Pre-Ingest File Submission Guidelines

Sproger B.: 'Technical Note Pre-Ingest File Submission Guidelines', October 2010

ECP-2008-DILI-518001

BHL-Europe

Technical Note

Pre-Ingest File Submission Guidelines

Deliverable number	<i>TN-SPRINT03-314</i>
Dissemination level	<i>Public</i>
Delivery date	<i>October 2010</i>
Status	<i>Final</i>
Author(s)	<i>Bernd Sproger</i>



eContentplus

This project is funded under the *eContentplus* programme²⁶,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

²⁶ OJ L 79, 24.3.2005, p. 1.

Table of contents

1	DOCUMENT HISTORY	67
1.1	CONTRIBUTORS	67
1.2	REVISION HISTORY	67
1.3	DISTRIBUTION.....	67
2	FILE SUBMISSION GUIDLINES	68
2.1	GENERAL RULES FOR DIRECTORY AND FILE NAMES	68
2.2	DIRECTORY STRUCTURE AND FILENAME PATTERN	68
2.3	EXAMPLES	69
3	FAQ	70
4	OPTIONAL: PACKAGE INFORMATION METADATA.....	71

1 Document History

1.1 Contributors

A discussion about the specifications was initiated and the following persons provided input that was used for the present document.

Person	Partner
Bernd Sproger	AIT

1.2 Revision History

Revision Date	Author	Version	Change Reference & Summary
2010-09-28	Bernd Sproger	0.1	1. Draft
2010-11-25	Bernd Sproger	0.2	Added to FAQ
2011-02-15	Bernd Sproger	0.3	Added to FAQ: OAI harvesting of metadata
2011-05-03	Chris Sleep	Final	Dissemination changed to public for D3.7

1.3 Distribution

This document has been distributed to:

Group	Date of issue	Version
BHL-Europe content providers (bhl-e.cp@lists.hu-berlin.de)	2010-10-19	0.1
BHL tech group (bhle-tech@googlegroups.com)	2010-10-19	0.1
BHL-Europe WP3	2010-10-19	0.1
BHL WIKI (https://bhl.wikispaces.com/BHL-E_WP3_PREINGEST)	2010-10-19	0.1

2 File Submission Guidelines

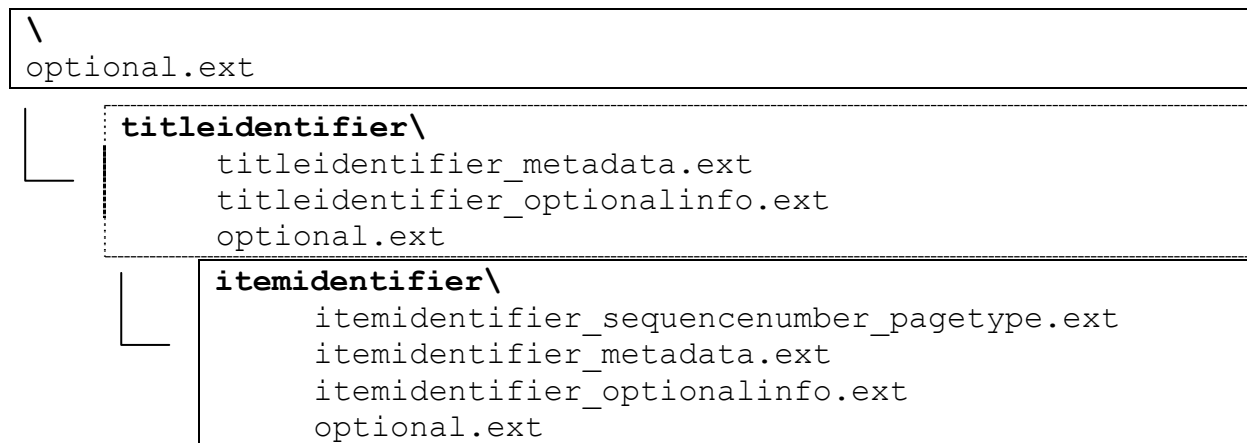
The file submission guidelines are necessary to allow automatic processing of submitted content and metadata. If you can't adhere to the file submission guidelines, please send an email to kochg@ait.co.at.

2.1 General Rules for Directory and File Names

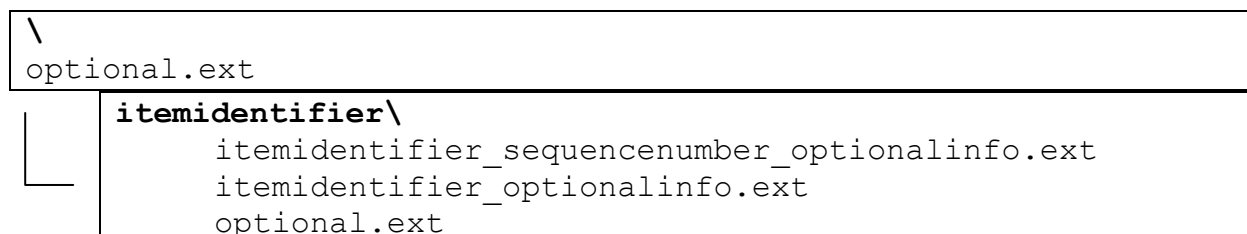
- Preferably use only ASCII characters and Western/Arabic numbers (0-9)
- Don't use: <, >, ", /, |, ?, *
- Avoid to use blank spaces
- Lower case file names are preferred ("abc" instead of "Abc")
- Use three-letter extensions for file names (.tif for TIFF images, .jpg for JPEGs, etc.)

2.2 Directory Structure and Filename Pattern

The directory structure aims to reliably identify titles and items and to reliably identify what files belong to which title or which item. You can create multiple folders for your items and multiple folders for your titles. Item folders are nested inside a title folder. Scanned images always belong to items:



If an item has no title, the titleidentifier directory must be omitted:



2.3 Examples

An item consists of a directory with multiple image files and one metadata file:

```
301519\  
  301519_0001.tif  
  301519_0002.tif  
  301519_metadata.xml
```

An item consisting of a directory with multiple image files, metadata files, and others:

```
301519\  
  301519_0001_cover.tif  
  301519_0002_blank.tif  
  301519_0003_title.tif  
  [...]  
  301519_0899.tif  
  301519_metadata.mrc  
  301519_article_01_metadata.mrc  
  301519_article_02_metadata.mrc  
  301519_articles_start_end_fileinfo.xml  
  301519_scanning_info.xml  
  301519_document_ocr.pdf  
  general_info.txt  
  thumbs.db  
  bag-info.txt
```

A title may consist of multiple items. Items could contain other items such as articles:

```
gartenkal\  
gartenkal_title.xml  
  821\  
    821_0001_cover.tif  
    821_0002_blank.tif  
    821_0003_title.tif  
    [...]  
    821_1002_page_999.tif  
    821_metadata.mrc  
    821_scanning_info.xml  
  831\  
    [...]  
    831_01\ [...]  
    831_02\ [...]  
  834\  
    [...]  
    834_01\ [...]  
    834_02\ [...]
```

Files that don't belong to titles or items need to be part of the top-level directory:

```

\  

general_info.txt  

all_metadata_database.sql  

  gartenkal\  

  gartenkal.mrc  

    821\  

      [...]  

    831\  

      [...]  

    834\  

      [...]  

  magnaturkdehelvet\  

  magnaturkdehelvet.mrc  

    871\  

      [...]  

    881\  

      [...]  

    891\  

      [...]
```

Example of a title where there are no separate item metadata files:

```

gartenkal\  

gartenkal_title_and_all_items_metadata.xml  

  821\  

    821_0001_cover.tif  

    821_0002_blank.tif  

    821_0003_title.tif  

    [...]  

    821_1002_page_999.tif  

  831\  

    [...]
```

3 FAQ

- My local identifiers (URNs) are using special characters which can't be used in filenames or directory names. What should I do?

We understand that URNs are a popular way to organise identifiers because they add namespaces to make identifiers more globally unique. URNs contain “:” which can't be used on Windows filesystems. Please exclude the URN prefix containing your namespace, e.g.: urn:nbn:de:gbv:089-3321752945 becomes 089-3321752945. Please always inform us about such steps, so we can correctly process your submissions afterwards.

- How do I need to structure articles, items and title metadata?

Put bibliographic metadata inside directories that hold scanned items such as :

```
NRBGE_0004\
NRBGE_0004\NRBGE_0004_Vol4.mrc
NRBGE_0004\NRBGE_0004_1905_0016\
NRBGE_0004\NRBGE_0004_1905_0016\NRBGE_0004_1905_0016_Issue16.mrc
NRBGE_0004\NRBGE_0004_1905_0016\NRBGE_0004_1905_0016_ArticlesForIssue16.m
```

- How do I proceed with submitting content if my metadata is available via OAI provider?

We gladly harvest your metadata via an OAI-PMH interface. Of course we still need means to identify which submitted content belongs to which metadata. Therefore it's recommended that you name your folders using the same identifiers that are used by your OAI provider. Please contact us for special cases where your OAI identifier can't be used.

4 Optional: Package Information Metadata

Attention: Following information is optional only and the intended audience are content providers with a strong technical background.

Submissions are going to be packaged once they are uploaded, and package information metadata will be generated and included. This prevents unwanted modification of the package payload (submitted content and metadata) and enables further processing of data and enrichment of metadata. We are using a software library developed by the Library of Congress called BagIt²⁷. If you or your digitisation provider are able to include package information metadata with your submission, please add a file called "bag-info.txt" to your items directory containing:

<Element>: <Content>

```
Source-Organization: AIT Angewandte Informationstechnik
Organization-Address: Klosterwiesgasse 33, Graz
Contact-Name: Dr. Walter Koch
Contact-Phone: +43 316-835359-74
Contact-Email: admin@ait.co.at
External-Description: old german book in Gothic print
Bagging-Date: 2010-10-01
External-Identifier: cat00098
Bag-Size: 2 GB
```

For your convenience we append the section 4.2 about BagIt metadata extracted from the BagIt v0.96 specification paper:

²⁷ <https://confluence.ucop.edu/display/Curation/BagIt>

4.2. Other bag metadata: bag-info.txt

The "bag-info.txt" file is a tag file that contains metadata elements describing the bag and the payload. The metadata elements contained in the "bag-info.txt" file are intended primarily for human readability. All metadata elements are optional. A metadata element consists of a label, a colon, and a value. Whitespace after the first non-whitespace in the value is considered part of the value. Long values may be folded (continued) onto the next line by inserting a newline (LF), a carriage return (CR), or carriage return plus newline (CRLF) and indenting the next line (any combination of spaces and tabs). It is recommended that lines not exceed 79 characters in length. Reserved metadata element names are case-insensitive and defined as follows.

Source-Organization

Organization transferring the content.

Organization-Address

Mailing address of the organization.

Contact-Name

Person at the source organization who is responsible for the content transfer.

Contact-Phone

International format telephone number of person or position responsible.

Contact-Email

Fully qualified email address of person or position responsible.

External-Description

A brief explanation of the contents and provenance.

Bagging-Date

Date (YYYY-MM-DD) that the content was prepared for delivery.

External-Identifier

A sender-supplied identifier for the bag.

Bag-Size

Size or approximate size of the bag being transferred, followed by an abbreviation such as MB (megabytes), GB, or TB; for example, 42600 MB, 42.6 GB, or .043 TB. Compared to Payload-Oxum (described next), Bag-Size is intended for human consumption.

Payload-Oxum

The "octetstream sum" of the payload, namely, a two-part number of the form "OctetCount.StreamCount", where OctetCount is the total number of octets (8-bit bytes) across all payload file content and StreamCount is the total number of payload files. Payload-Oxum is easy to compute (e.g., on Unix "wc -lc `find data/ -type f`") and should be included in "bag-info.txt" if at all possible. Compared to Bag-Size (above), Payload-Oxum is intended for machine consumption.

Bag-Group-Identifier

A sender-supplied identifier for the set, if any, of bags to which it logically belongs. This identifier must be unique across the sender's content, and if recognizable as belonging to a globally unique scheme, the receiver should make an effort to honor reference to it.

Bag-Count

Two numbers separated by "of", in particular, "N of T", where T is the total number of bags in a group of bags and N is the ordinal number within the group; if T is not known, specify it as "?" (question mark). Examples: 1 of 2, 4 of 4, 3 of ?, 89 of 145.

Internal-Sender-Identifier

An alternate sender-specific identifier for the content and/or bag.

Internal-Sender-Description

A sender-local prose description of the contents of the bag.

In addition to these metadata elements, other arbitrary metadata elements may also be present.

III.II Technote: WebDAV

Sproger B.: “BHL-Europe WebDAV fallback solution to replace FTPS service”, 2011-01-03

ECP-2008-DILI-518001

BHL-Europe

Technical Note

BHL-Europe WebDAV fallback solution to replace FTPS service

Deliverable number	<i>TN-Addendum-WebDAV</i>
Dissemination level	<i>Public</i>
Delivery date	<i>2011-01-03</i>
Status	<i>Final</i>
Author(s)	<i>Bernd Sproger</i>



eContentplus

This project is funded under the *eContentplus* programme²⁸,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

²⁸ OJ L 79, 24.3.2005, p. 1.

Table of contents

1	DOCUMENT HISTORY	77
1.1	CONTRIBUTORS	77
1.2	REVISION HISTORY	77
1.3	DISTRIBUTION.....	77
2	PURPOSE OF THIS DOCUMENT.....	78
3	SUPPORTED WEBDAV CLIENTS.....	78
4	LOGIN CREDENTIALS FOR WEBDAV.....	78
5	SETUP GUIDES.....	78
5.1	BITKINEX QUICK SETUP GUIDE.....	78
5.2	CYBERDUCK QUICK SETUP GUIDE.....	80
5.3	UBUNTU QUICK SETUP GUIDE	81

1 Document History

1.1 Contributors

A discussion about this issue was initiated and the following persons provided input that was used for the present document.

Person	Partner
Bernd Sproger	AIT
Antonio G Valdecasas	CSIC
Tom Gilissen	Naturalis

1.2 Revision History

Revision Date	Author	Version	Change Reference & Summary
2010-12-21	Bernd Sproger	0.1	1. Draft
2011-01-03	Bernd Sproger	0.2	1. Revision, added Ubuntu Setup Guide
2011-05-03	Chris Sleep	Final	Dissemination changed to public for D3.7

1.3 Distribution

This document has been distributed to:

Group	Date of issue	Version
Naturalis, RMCA, MNHN, CSIC, RBGE	2010-12-21	0.1

2 Purpose of this document

This document describes a fallback solution to upload content and metadata to servers at NHM by using secured WebDAV instead of FTPS. WebDAV is similar to FTPS but uses HTTP as transmission protocol and therefore should be less error prone to very restrictive firewall policies at content provider's sites. There come certain downfalls with the use of WebDAV such as less file transmission speed, but anyway files are transmitted reliably.

3 Supported WebDAV clients

We have done file transmission tests using the following clients:

	<u>Operating Systems</u>	<u>Source</u>
BitKinex	Windows	http://www.bitkinex.com/
Cyberduck	Windows / MacOSX	http://cyberduck.ch/
cadaver	Linux	http://www.webdav.org/cadaver/

4 Login Credentials for WebDAV

Since WebDAV is a fallback solution for FTPS we are using the same login credentials which have already been provided to content providers.

5 Setup Guides

Setup and usage of WebDAV clients is generally very similar to usage of FTP clients such as Filezilla. We have provided a few screenshots and setup steps for BitKinex and Cyberduck to help you through the first few steps. File upload, download, etc. is identical with FTP clients.

5.1 *BitKinex Quick Setup Guide*

BitKinex is an advanced WebDAV client and supports many different settings. We have compiled a few quick setup steps.

- 1) In the Main Menu go to 'Data Source' -> 'New' -> 'Http/WebDAV'
- 2) Name the new entry e.g. 'NHM BHL-Europe'
- 3) Next a window pops up giving you many configuration options (see Figure 5-1).

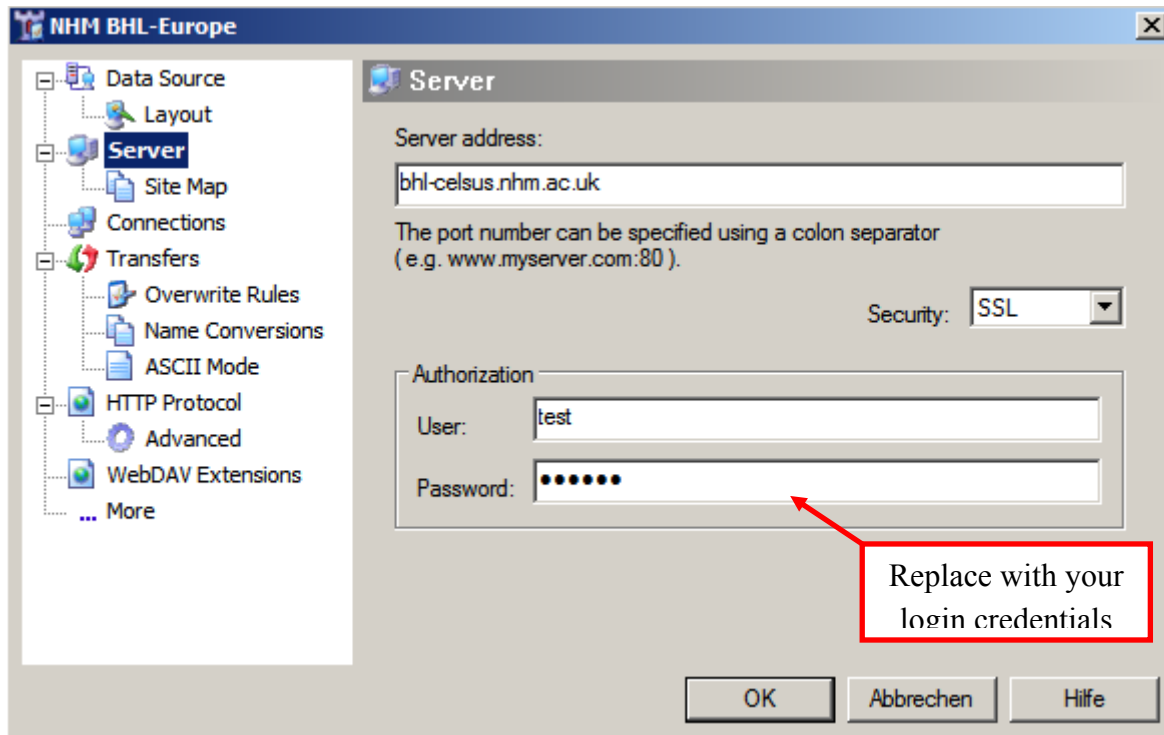


Figure 5-1: Server configuration window of BitKinex client

- 3a) Select 'Server' and enter 'bhl-celsus.nhm.ac.uk' in the field 'Server address'
- 3b) Choose 'SSL' in the Security dropdownbox.
- 3c) Enter the fields 'User' and 'Password' using your FTPS login credentials already provided to you by NHM.
- 4) Select 'Site Map' in the left menu (see Figure 5-2)

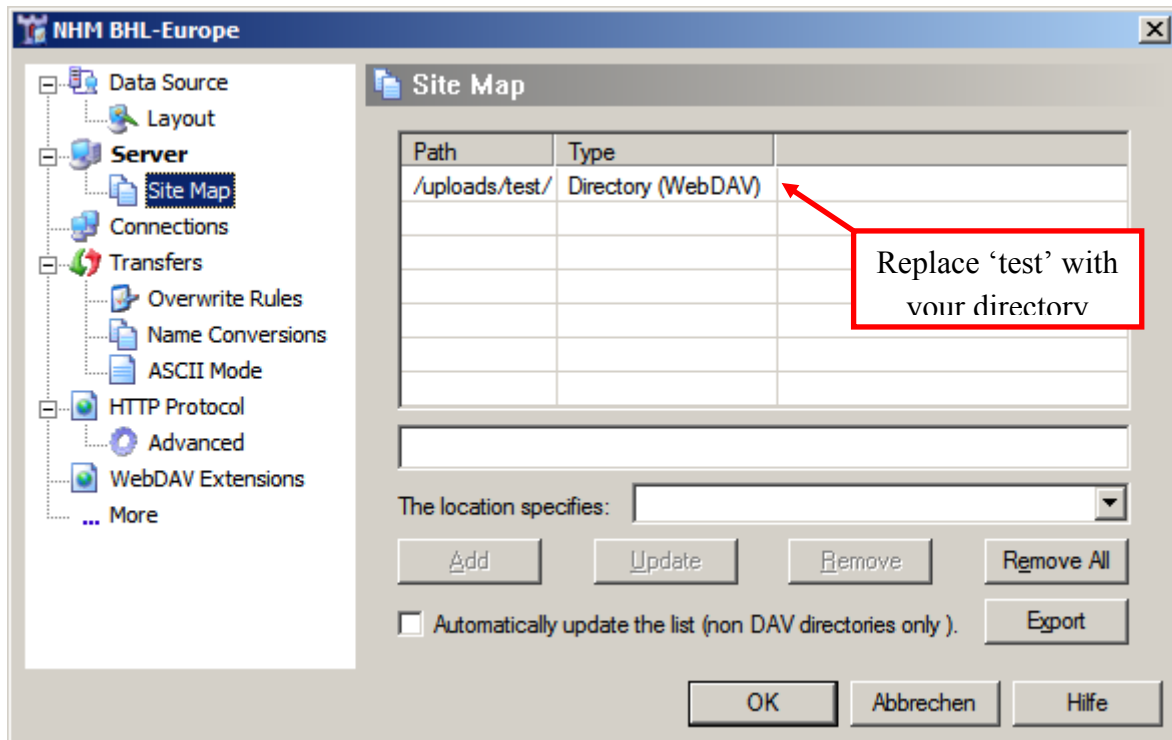


Figure 5-2: Site Map configuration window

- 4a) Click on the the row starting with '/' thus highlighting it.
- 4b) Right under the area with the cells edit the field containg '/' and replace it with e.g. '/uploads/NAT/'
- 4c) Click on the button 'Update'
- 5) Click on the button OK

5.2 Cyberduck Quick Setup Guide

Cyberduck has been tested on Windows only, but usage for MacOSX should be very similar since this client originally was published for MacOSX:

- 1) Select 'New Connection'
- 2) Enter the connection details as seen in Figure 5-3.
- 3) Double-click the newly created connection to login
- 4) Enter your password and save it for future use.

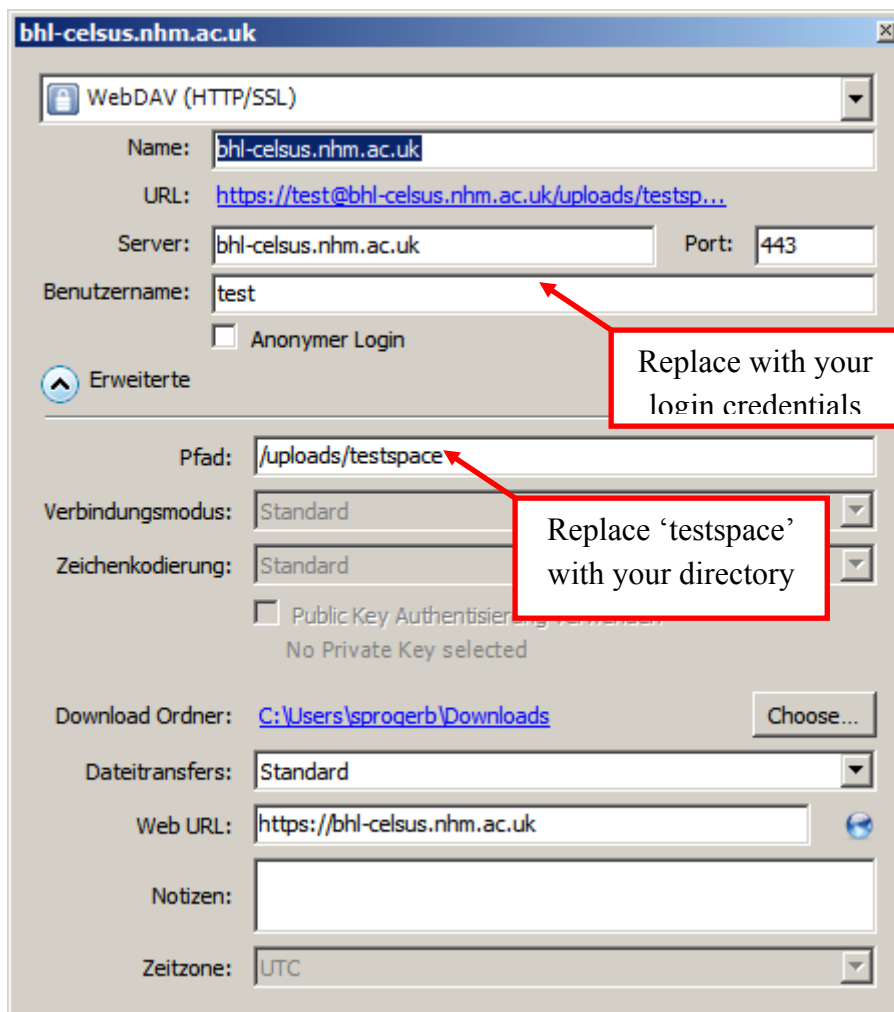


Figure 5-3: New Connection configuration window

5.3 *Ubuntu Quick Setup Guide*²⁹

Taken out of the online Ubuntu manual for your convenience:

(Ubuntu Documentation > Ubuntu 10.10 > Internet and Networks > Connect to a server > WebDAV (HTTP))

Click Places → Connect to Server....

From Service Type choose WebDAV (HTTP).

Enter the server address in Server.

Enter Port and Folder and Username if you need to, this is optional.

If you want a bookmark, click the checkbox and enter a bookmark name.

Click Connect.

You will be prompted for a password if necessary.

²⁹ <https://help.ubuntu.com/10.10/internet/C/connecttoserver-webdav.html>

III.III Technote: BHL-Europe GUID

Sproger B., Koch W.: “ BHL-Europe GUID Architecture and Implementation Approach”,
2011-01-20

ECP-2008-DILI-518001

BHL-Europe

Technical Note

BHL-Europe GUID Architecture and Implementation Approach

Deliverable number	<i>TN-Addendum-GUID</i>
Dissemination level	<i>Public</i>
Delivery date	<i>2011-01-20</i>
Status	<i>Final</i>
Author(s)	<i>Bernd Sproger</i>



eContentplus

This project is funded under the *eContentplus* programme³⁰,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

³⁰ OJ L 79, 24.3.2005, p. 1.

Table of contents

1	DOCUMENT HISTORY	85
1.1	CONTRIBUTORS	85
1.2	REVISION HISTORY	85
1.3	DISTRIBUTION.....	85
2	PURPOSE OF THIS DOCUMENT.....	86
3	ARCHITECTURE OF BHL-EUROPE GUIDS	86
4	IMPLEMENTATION ASPECTS OF IDENTIFIERS.....	87
5	CONCLUSION	87

1 Document History

1.1 Contributors

A discussion about this issue was initiated and the following persons provided input that was used for the present document.

Person	Partner
Bernd Sproger, Walter Koch	AIT

1.2 Revision History

Revision Date	Author	Version	Change Reference & Summary
2011-01-20	Bernd Sproger	0.1	1. Draft
2011-05-03	Chris Sleep	Final	Dissemination changed to public for D3.7

1.3 Distribution

This document has been distributed to:

Group	Date of issue	Version

2 Purpose of this document

This document describes the general BHL-Europe GUID architecture and implementation approach. It is closely based on a publication of Adam J. Smith³¹ where Handle system³² based webservices are implemented at the University of Cornell. Handle system allows for “efficient, extensible, and secure resolution services for unique and persistent identifiers of digital objects”. Unique identifiers are assigned based on the NOID³³ (Nice Opaque Identifier) specification. Webservices are designed and implemented to enable CRUD (create, read, update, delete) for GUIDs.

3 Architecture of BHL-Europe GUIDs

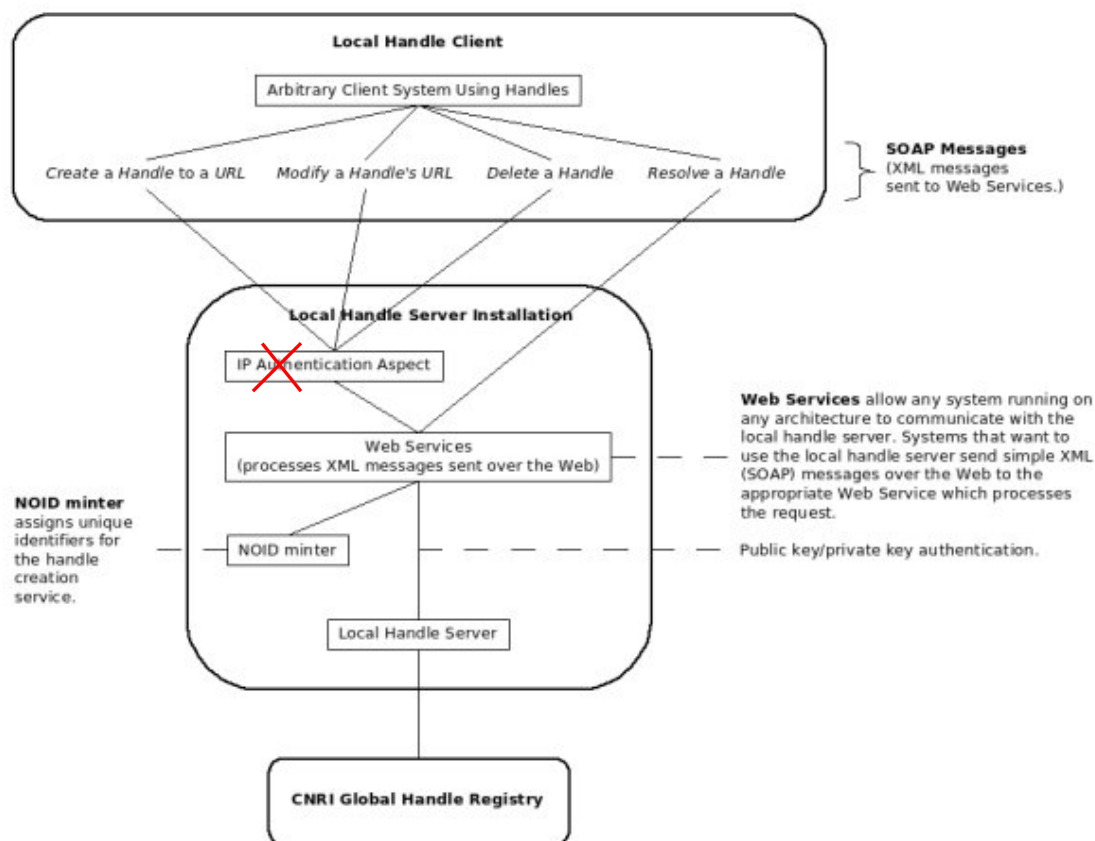


Figure 3-1: Cornell University's architecture of web services integrating local handle client and server and NOID minter. Modified for BHL-Europe.

³¹ Adam J. Smith: "Developing Handle System® Web Services at Cornell University", D-Lib Magazine September/October 2007, Volume 13 Number 9/10, ISSN 1082-9873.

<http://www.dlib.org/dlib/september07/smith/09smith.html>

³² <http://www.handle.net>

³³ <https://confluence.ucop.edu/display/Curation/NOID>

As you can see in Figure 3-1 Cornell University's architecture is perfectly suited for implementation in the BHL-Europe context with small amendements. It's a simple web service based design that's using the local handle client and handle server Java implementations provided by Handle.NET. The NOID minter is using a Perl tool³⁴ to mint identifiers which is also providing a CGI-based interface to be easily integrated into a service oriented architecture.

4 Implementation Aspects of Identifiers

Example for a GUID using the already registered BHL-Europe Handle³⁵:

```
hdl:10706/f5td9v7s
```

```
hdl:<handle identifier>/<NOID>
```

The NOID wiki sums up some very important aspects about the design of an identifier:

“To help stability, an opaque identifier doesn't contain any information related to potentially changeable properties. For instance, if an identifier contains an organizational acronym and that organization is merged with another, there is often political pressure to break with the past, which means pressure not to support previously published identifiers in which the old acronym appears. Opaque identifiers also have the advantage that they can be short; for example, using combinations of letters and digits, only four characters are needed to represent as many as 1.6 million identifiers.”

Therefore identifiers mustn't contain any 'human-readable' contents whatsoever! AIT particularly stresses this fact since it has been subject of discussion³⁶ in the past to include changeable properties in the design of identifiers. We also suggest reading of the ARK³⁷ specification, specifically sections '2.4 The Name Part', '3 Naming Considerations', '3.3 Names are Political, not Technological'. These snippets of the ARK specifications also explain why identifiers mustn't include any specific object attributes.

5 Conclusion

AIT supports the idea of using GUIDs for BHL content and provides an approach of how to design and implement a solution based on already established best-practice approaches.

³⁴ <http://search.cpan.org/dist/Noid/>

³⁵ https://bhl.wikispaces.com/Installation_of_Handle.net_server

³⁶ Based on a presentation given by Lee Namba during the BHL-Europe Tech Group Meeting in Amsterdam in 2010-10.

³⁷ <https://confluence.ucop.edu/display/Curation/ARK>

III.IV Technote: Dataflow

Sproger B.: “BHL-Europe DataFlow from BHL-US to Europeana EXAMPLE”, 2010-09-27

ECP-2008-DILI-518001

BHL-Europe

Technical Note

**BHL-Europe DataFlow from BHL-US to
Europeana
EXAMPLE**

Deliverable number	<i>TN-Addendum-DataFlow</i>
Dissemination level	<i>Public</i>
Delivery date	<i>2010-09-27</i>
Status	<i>Final</i>
Author(s)	<i>Bernd Sproger</i>



eContentplus

This project is funded under the *eContentplus* programme³⁸,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

³⁸ OJ L 79, 24.3.2005, p. 1.



Table of contents

1	DOCUMENT HISTORY	91
1.1	CONTRIBUTORS	91
1.2	REVISION HISTORY	91
1.3	DISTRIBUTION.....	91
2	PURPOSE OF THIS DOCUMENT.....	92
3	EXAMPLE DATAFLOW FROM BHL-US.....	92
3.1	A1 - PROVIDES BHL-US METADATA	92
3.2	A2 - FETCHES BHL-US METADATA RECORDS	92
3.3	A3 - MAPS FROM BHL-US TO ESE	93
3.4	A4 - PROVIDES ESE METADATA	95
3.5	A5 - FETCHES ESE METADATA RECORDS	95
3.6	A6 - DISPLAYS METADATA RECORDS.....	95
4	'THROW-AWAY' BHL-EUROPE PROTOTYPE.....	97
5	SUGGESTED ARCHITECTURAL APPROACH TO DATA MAPPING	97
6	CONCLUSION	97
7	APPENDIX	97
7.1	IDEF0 DIAGRAMS	97
7.2	BHL-US HTTP REST API RESPONSE FOR GETTITLEMETADATA()	102
7.3	MAPPED METADATA USING THE EUROPEANA ESE SCHEMA	103

1 Document History

1.1 Contributors

A discussion about this issue was initiated and the following persons provided input that was used for the present document.

Person	Partner
Bernd Sproger	AIT

1.2 Revision History

Revision Date	Author	Version	Change Reference & Summary
2010-09-09	Bernd Sproger	0.1	1. Draft
2010-09-15	Bernd Sproger	0.2	1. Final
2010-09-27	Bernd Sproger	0.3	1. Revision
2011-05-03	Chris Sleep	Final	Dissemination changed to public for D3.7

1.3 Distribution

This document has been distributed to:

Group	Date of issue	Version
MFN Berlin	2010-09-09	0.1
BHL-Europe Tech Group	2010-09-15	0.2
BHL-Europe Tech Group	2010-09-27	0.3

2 Purpose of this document

This document describes in detail the implemented BHL-Europe dataflow, i.e. transfer and mapping of metadata from BHL-US to Europeana. Title and item metadata is fetched from BHL-US, afterwards mapped to ESE, and eventually published via OAI-PMH to be harvested by Europeana.

3 Example DataFlow from BHL-US

The following example shows the data flow from BHL-US to Europeana via the BHL-Europe prototype portal. We are referencing the IDEF0³⁹ diagram boxes A1-A6 which are shown on the IDEF0 diagram “*BHL-Europe Data Flow with regards to BHL-US and ESE (AIT, 2010-09-07) (Old)*”. This diagram shows the dataflow without use of the BHL-Europe MODS schema since this schema isn’t yet available at the time this document is written. The IDEF0 diagram “*BHL-Europe Data Flow with regards to BHL-US, BHL-Europe MODS schema, EDM/ESE (AIT, 2010-09-07)*” is also included in this document for the sake of completeness, but is not explicitly referenced in the following chapters.

3.1 A1 - Provides BHL-US Metadata

BHL-US provides publicly available information about the BHL-US application programming interfaces (API). OAI-PMH, REST, SOAP are available.⁴⁰

3.2 A2 - Fetches BHL-US Metadata Records

AIT fetches a BHL-US metadata for Title ID 2 “*Examen classis dioeciae*” using the BHL-US API.⁴¹ Full response is listed in the Appendix. For reasons of simplicity we have extracted the following text from the BHL-US API response which shows how authors are structured.

```

= <Authors>
= <Creator>
  <CreatorID>1106</CreatorID>
  <Name>East India Company.</Name>
  <Numeration />
  <Unit>Museum.</Unit>
  <Title />
  <Location />
  <Dates />
</Creator>
= <Creator>
  <CreatorID>4</CreatorID>
  <Name>Fleischer, Max,</Name>
  <Numeration />
  <Unit />

```

³⁹ IDEF Integrated Definition Methods, <http://www.idef.com/>

⁴⁰ API Documentation, <http://www.biodiversitylibrary.org/api2/docs/docs.html>

⁴¹ Request (API Key is confidential):

<http://www.biodiversitylibrary.org/api2/httpquery.ashx?op=GetTitleMetadata&titleid=2&items=t&apikey=f250642e-fa4c-4e36-9a56-c164e5681ddf>

```

<Title />
<Location />
<Dates>1861-1930.</Dates>
</Creator>
= <Creator>
<CreatorID>3229</CreatorID>
<Name>Kjellenberg, Fredrik Ulrik,</Name>
<Numeration />
<Unit />
<Title />
<Location />
<Dates>1795-1862.</Dates>
</Creator>
= <Creator>
<CreatorID>3249</CreatorID>
<Name>Söderberg, Christopher,</Name>
<Numeration />
<Unit />
<Title />
<Location />
<Dates>1804-1833.</Dates>
</Creator>
= <Creator>
<CreatorID>108</CreatorID>
<Name>Thunberg, Carl Peter,</Name>
<Numeration />
<Unit />
<Title />
<Location />
<Dates>1743-1828.</Dates>
</Creator>
= <Creator>
<CreatorID>112</CreatorID>
<Name>Wallich, N.</Name>
<Numeration />
<Unit />
<Title />
<Location />
<Dates>1786-1854.</Dates>
</Creator>
</Authors>

```

3.3 A3 - Maps from BHL-US to ESE

AIT maps the received API response to Europeana ESE schema using the BHL-Europe prototype portal. The following screenshots represent a search conducted on <http://bhl.ait.co.at> using the search terms “*Examen classis dioeciae*”.

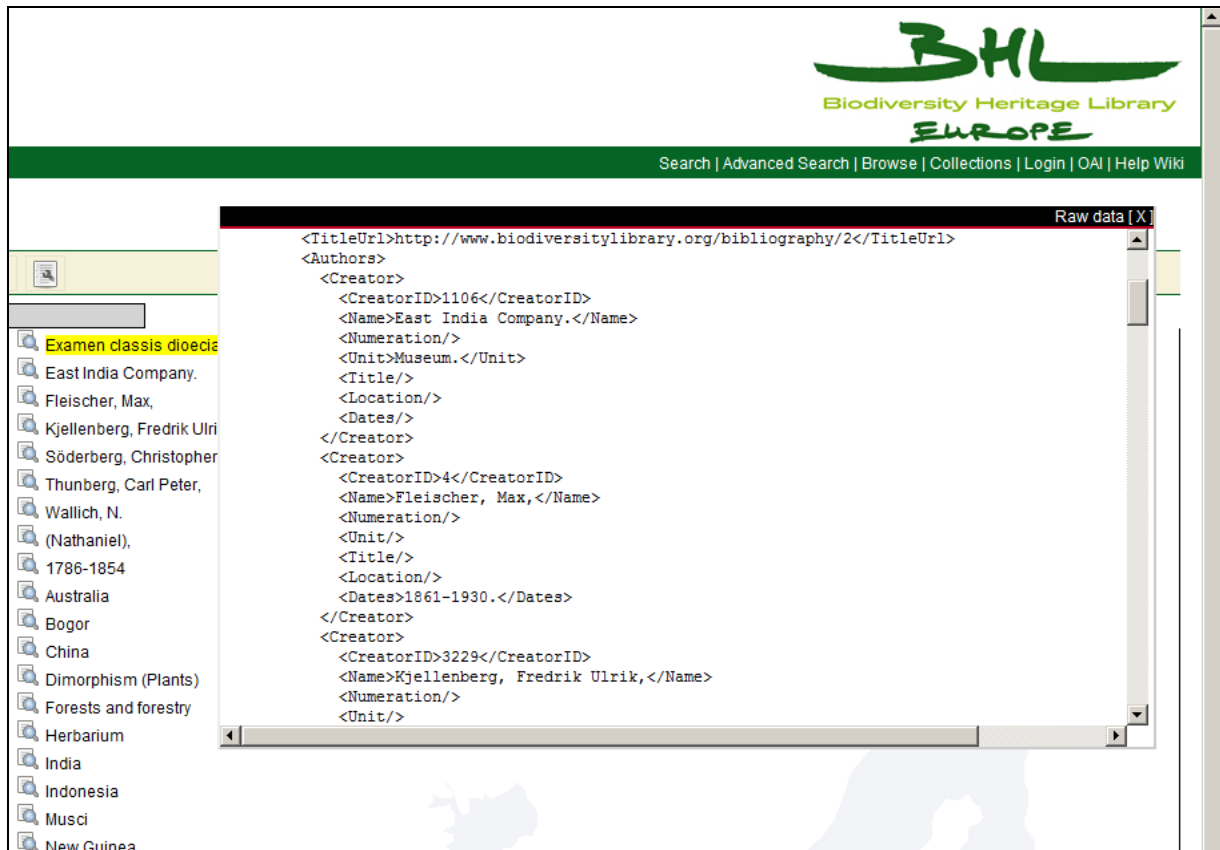
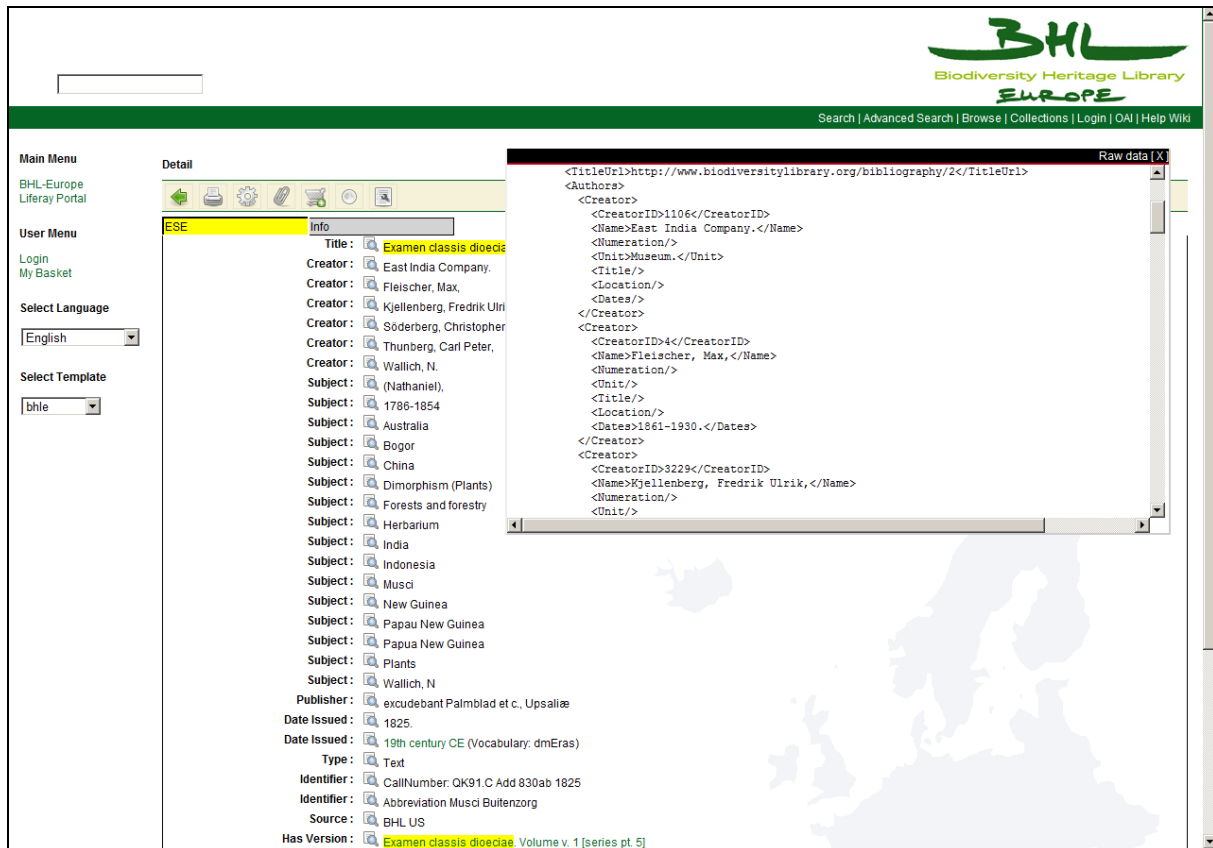


Figure 3-1: Zoomed-in screenshot from <http://bhl.ait.co.at> showing raw data received via BHL-US API.



The screenshot shows the BHL Europe Liferay Portal interface. On the left is a navigation menu with options like 'Main Menu', 'User Menu', 'Login', 'My Basket', 'Select Language' (set to English), and 'Select Template' (set to bhle). The main content area is titled 'Detail' and shows metadata for the record 'Examen classis dioeciae'. The metadata includes fields for Title, Creator (listing East India Company, Fleischer, Max, Kjellenberg, Fredrik Ulrik, Söderberg, Christopher, Thunberg, Carl Peter, and Wallich, N.), Subject (listing various geographical locations and botanical topics), Publisher, Date Issued (1825), Type (Text), Identifier, and Source (BHL US). A 'Raw data' window is open, displaying the XML response from the BHL-US API, which includes the same metadata fields in a structured format.

Figure 3-2: Fullscreen screenshot from <http://bhl.ait.co.at> showing Raw Data received via BHL-US API.

3.4 A4 - Provides ESE Metadata

AIT provides mapped metadata fields for Europeana using an OAI-PMH provider. Fulltext of the mapped metadata is listed in the Appendix. For reasons of simplicity we have extracted authors/creators from the BHL-US metadata which were mapped to ESE by the BHL-Europe German prototype.

```

<ese-Creator>East India Company.</ese-Creator>
<ese-Creator>Fleischer, Max.</ese-Creator>
<ese-Creator>Kjellenberg, Fredrik Ulrik,</ese-Creator>
<ese-Creator>Söderberg, Christopher,</ese-Creator>
<ese-Creator>Thunberg, Carl Peter,</ese-Creator>
<ese-Creator>Wallich, N.</ese-Creator>

```

3.5 A5 - Fetches ESE Metadata Records

Europeana harvests ESE metadata records provided by AIT via OAI-PMH harvester.

3.6 A6 - Displays Metadata Records

Europeana displays the fetched metadata records. We have made screenshots to show that the metadata fields which were retrieved by the BHL-US API response, are now shown on Europeana.

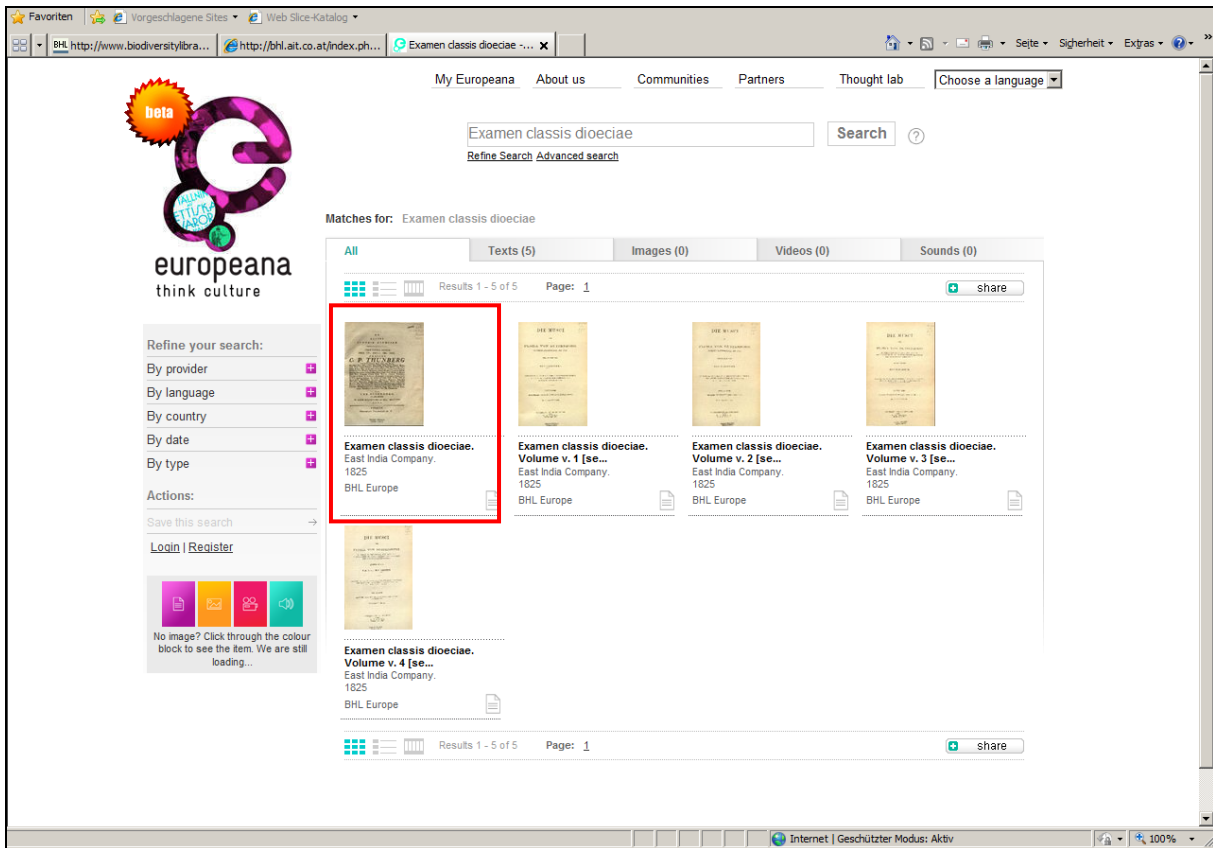


Figure 3-3: Screenshot showing search results for “Examen classis dioeciae” on <http://www.europeana.eu>.

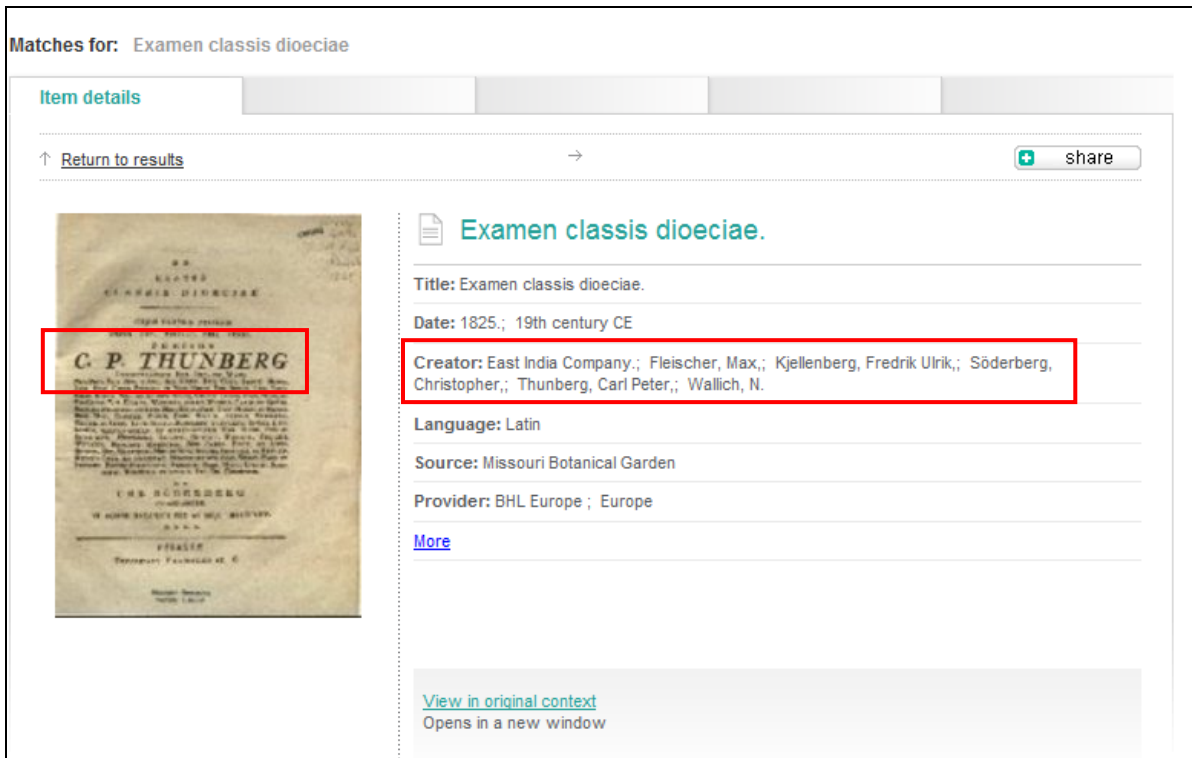


Figure 3-4: Screenshot showing detailed result for “Examen classis dioeciae”.

4 ‘Throw-Away’ BHL-Europe Prototype

The current BHL-Europe prototype which is due by M18 is based on previous developments and adaptations of the DISMARC search portal. It’s important to emphasize that the prototype won’t be developed any further. Instead it will be broken down into several small components so that we can create micro-services to be used for the BHL-Europe community portal. The current BHL-Europe prototype is implemented in PHP5. All mappings are done via DOM⁴² manipulation of incoming XML-formatted metadata. There is a separate mapping class for each metadata provider such as BHL-US which implements the specific requirements to map to ESE. ESE is used as the target schema. There are no other schemas available at the time. We are using Apache SOLR⁴³ to create the search index.

5 Suggested Architectural Approach to Data Mapping

The IDEF0 diagram “*BHL-Europe Data Flow with regards to BHL-US, Metadata Gateway, BHL-Europe MODS Schema, EDM/ESE (AIT, 2010-09-27)*” shows a new approach to separate the metadata mapping from the Pre-Ingest Tool. The introduced gateway component invokes a micro-service (web service) which is used to map the incoming data (e.g. BHL-US metadata) to the outgoing format (i.e. BHL-Europe MODS). The service is loosely coupled to the rest of the Pre-Ingest Tool, thus allowing to be exchanged easily. Therefore it will be possible to introduce new mappings as micro-services.

6 Conclusion

It’s important to emphasize that no metadata was lost or incorrectly mapped during the transfer of metadata fields from BHL-US to Europeana. Furthermore this technote demonstrates impressively that the quality of input data is crucial.

7 Appendix

7.1 IDEF0 diagrams

The following diagrams depict different versions of the dataflows:

- “*BHL-Europe Data Flow with regards to BHL-US and ESE (AIT, 2010-09-07) (Old)*”
- “*BHL-Europe Data Flow with regards to BHL-US, BHL-Europe MODS schema, EDM/ESE (AIT, 2010-09-07)*”
- “*BHL-Europe Data Flow with regards to BHL-US, Metadata Gateway, BHL-Europe MODS Schema, EDM/ESE (AIT, 2010-09-27)*”

The first dataflow was used to map BHL-US metadata to ESE. This dataflow is going to change with the introduction of the BHL-Europe MODS schema.

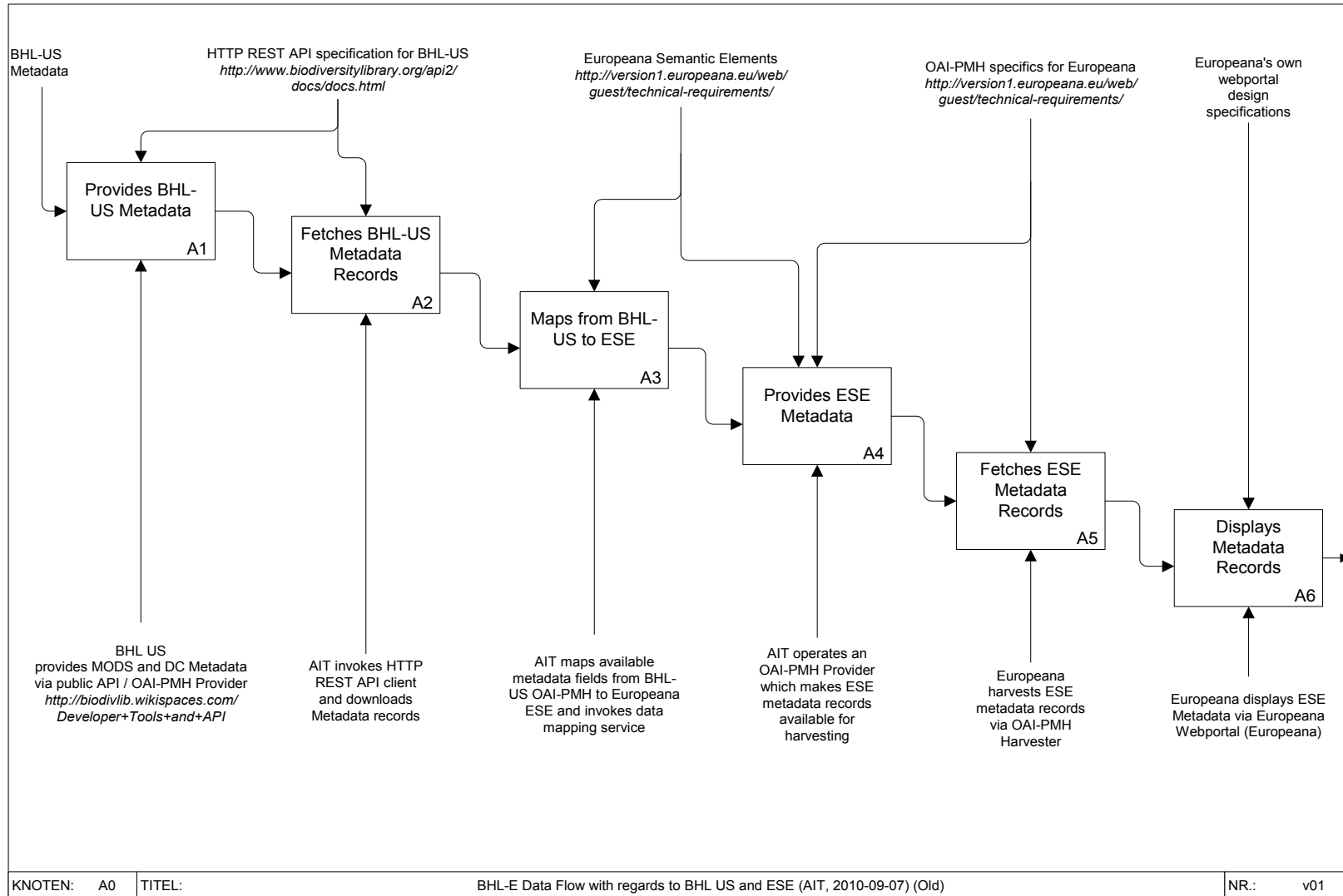
The second diagram demonstrates the dataflow once the BHL-Europe MODS schema is in use. This will also support the new EDM/ESE schema.

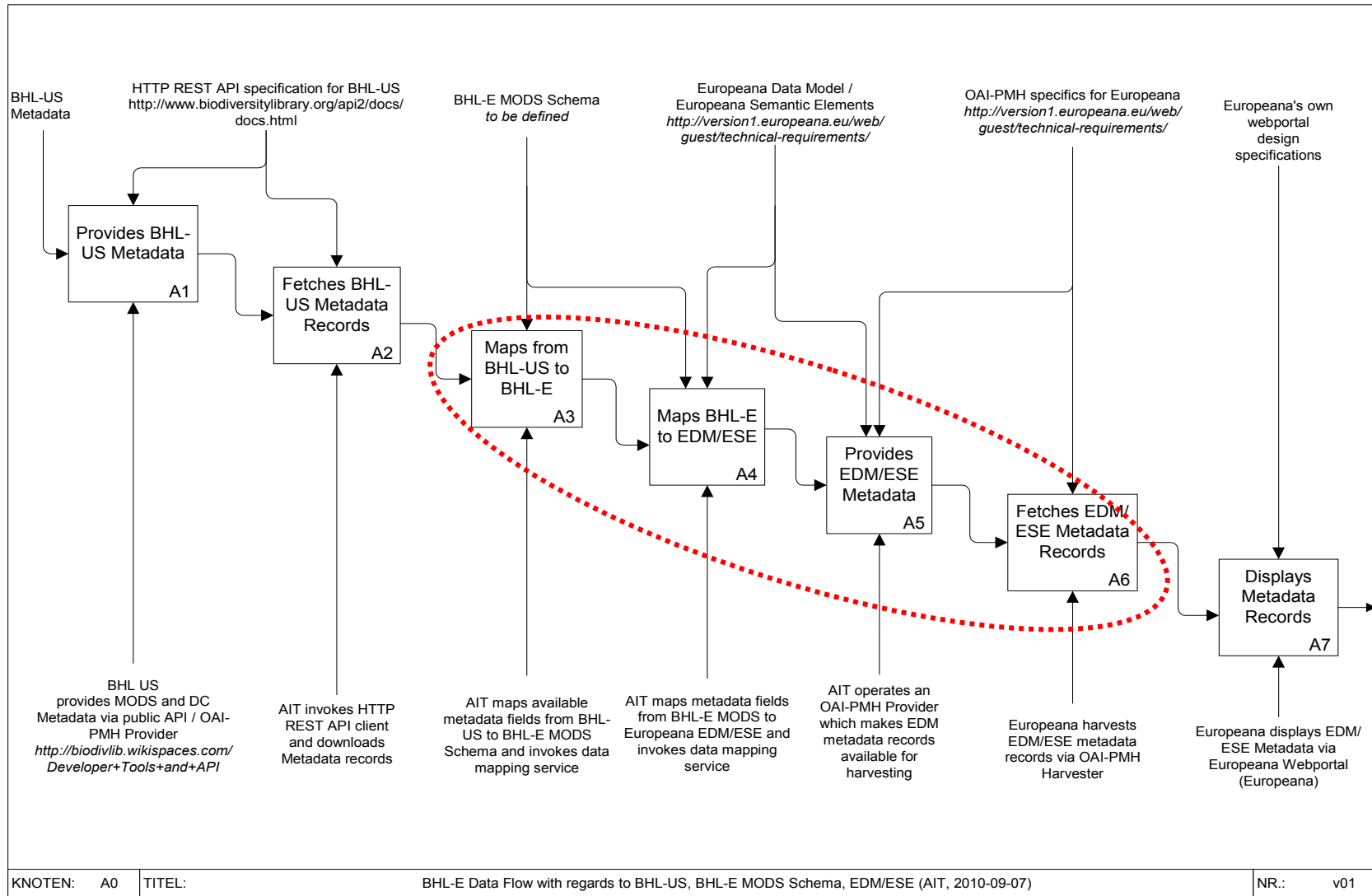
⁴² Document Object Model, <http://www.w3schools.com/dom/>

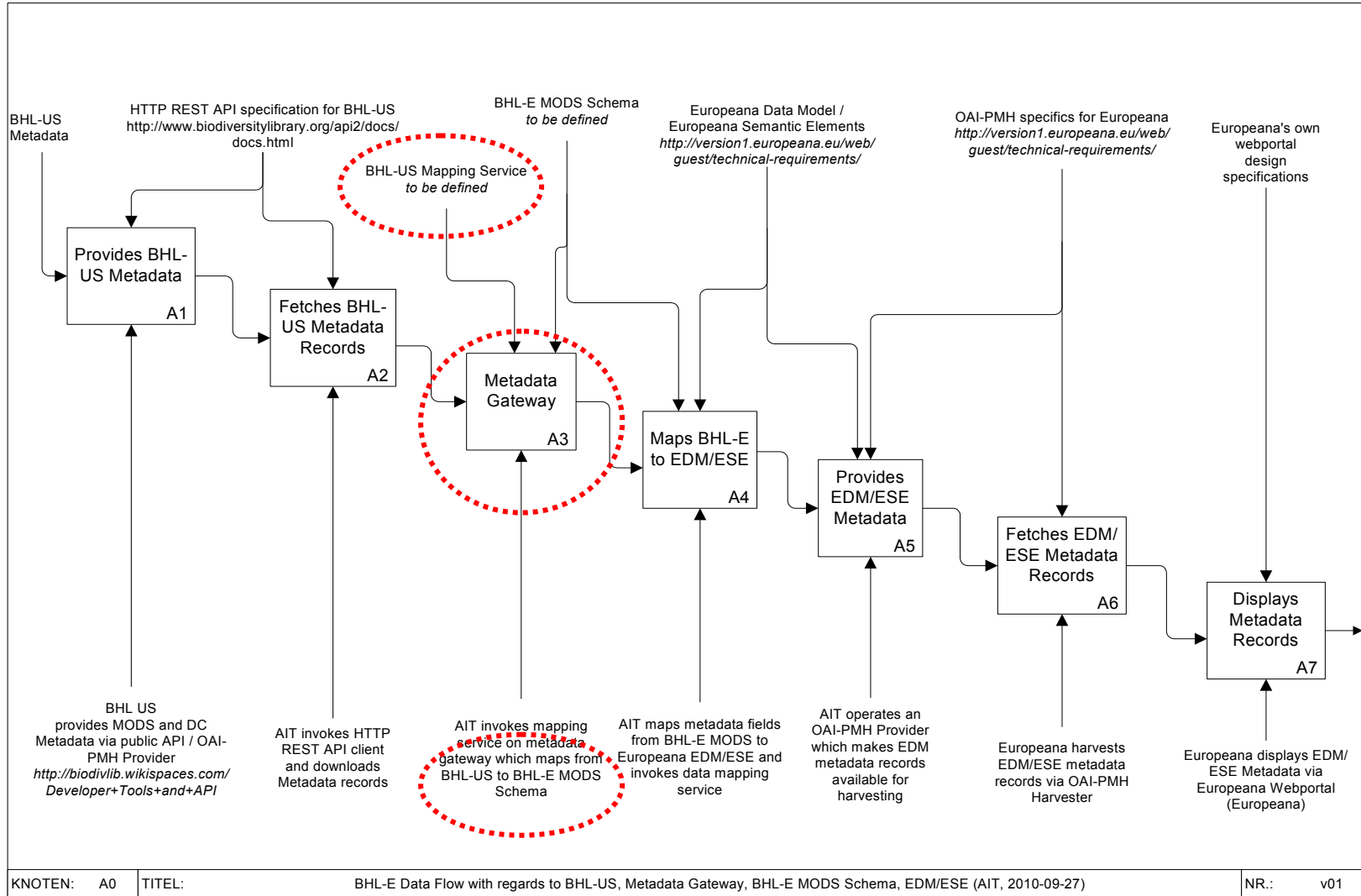
⁴³ <http://lucene.apache.org/solr/>



The third dataflow introduces a metadata gateway through which the metadata mapping will be decoupled from the dataflow. Therefore metadata mapping becomes even more transparent and future mappings can be integrated easier.







7.2 BHL-US HTTP REST API Response for GetTitleMetadata()

```

<?xml version="1.0" encoding="utf-8" ?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Status>ok</Status>
  <Result>
    <TitleID>2</TitleID>
    <FullTitle>Examen classis dioeciae.</FullTitle>
    <ShortTitle>Examen classis dioeciae.</ShortTitle>
    <SortTitle>Examen classis dioeciae.</SortTitle>
    <CallNumber>QK91.C Add 830ab 1825</CallNumber>
    <Edition />
    <PublisherPlace>Upsaliae :</PublisherPlace>
    <PublisherName>excudebant Palmblad et c.,</PublisherName>
    <PublicationDate>1825.</PublicationDate>
    <PublicationFrequency />
    <TitleUrl>http://www.biodiversitylibrary.org/bibliography/2</TitleUrl>
  </Result>
  <Authors>
    <Creator>
      <CreatorID>1106</CreatorID>
      <Name>East India Company.</Name>
      <Numeration />
      <Unit>Museum.</Unit>
      <Title />
      <Location />
      <Dates />
    </Creator>
    <Creator>
      <CreatorID>4</CreatorID>
      <Name>Fleischer, Max,</Name>
      <Numeration />
      <Unit />
      <Title />
      <Location />
      <Dates>1861-1930.</Dates>
    </Creator>
    <Creator>
      <CreatorID>3229</CreatorID>
      <Name>Kjellenberg, Fredrik Ulrik,</Name>
      <Numeration />
      <Unit />
      <Title />
      <Location />
      <Dates>1795-1862.</Dates>
    </Creator>
    <Creator>
      <CreatorID>3249</CreatorID>
      <Name>Söderberg, Christopher,</Name>
      <Numeration />
      <Unit />
      <Title />
      <Location />
      <Dates>1804-1833.</Dates>
    </Creator>
  </Authors>

```

```

=<Creator>
<CreatorID>108</CreatorID>
<Name>Thunberg, Carl Peter,</Name>
<Numeration />
<Unit />
<Title />
<Location />
<Dates>1743-1828.</Dates>
</Creator>
=<Creator>
<CreatorID>112</CreatorID>
<Name>Wallich, N.</Name>
<Numeration />
<Unit />
<Title />
<Location />
<Dates>1786-1854.</Dates>
</Creator>
</Authors>
+ <Subjects>
- <Identifiers>
+ <TitleIdentifier>
+ <TitleIdentifier>
+ <TitleIdentifier>
</Identifiers>
=<Items>
=<Item>
<ItemID>7</ItemID>
<PrimaryTitleID>2</PrimaryTitleID>
<ThumbnailPageID>600316</ThumbnailPageID>
<Source>Botanicus</Source>
<SourceIdentifier>31753002085667</SourceIdentifier>
<Volume>v. 1 [series pt. 5]</Volume>
<Contributor>Missouri Botanical Garden</Contributor>
<Sponsor>Missouri Botanical Garden</Sponsor>
<Language>Dutch</Language>
<LicenseUrl />
<Rights />
<DueDiligence />
<CopyrightStatus />
<CopyrightRegion />
<ItemUrl>http://www.biodiversitylibrary.org/item/7</ItemUrl>
<TitleUrl>http://www.biodiversitylibrary.org/bibliography/2</TitleUrl>
<ItemThumbUrl>http://www.biodiversitylibrary.org/pagethumb/600316</ItemThumbUrl>
</Item>
+ <Item>
+ <Item>
+ <Item>
+ <Item>
</Items>
</Result>
</Response>

```

7.3 Mapped metadata using the Europeana ESE schema

```

<?xml version="1.0" encoding="utf-8" ?>
=<regnet-document version="1.0">
=<section name="ESE">

```

```

<ese-Title>Examen classis dioeciae.</ese-Title>
<ese-Creator>East India Company.</ese-Creator>
<ese-Creator>Fleischer, Max,</ese-Creator>
<ese-Creator>Kjellenberg, Fredrik Ulrik,</ese-Creator>
<ese-Creator>Söderberg, Christopher,</ese-Creator>
<ese-Creator>Thunberg, Carl Peter,</ese-Creator>
<ese-Creator>Wallich, N.</ese-Creator>
<ese-Subject>(Nathaniel),</ese-Subject>
<ese-Subject>1786-1854</ese-Subject>
<ese-Subject>Australia</ese-Subject>
<ese-Subject>Bogor</ese-Subject>
<ese-Subject>China</ese-Subject>
<ese-Subject>Dimorphism (Plants)</ese-Subject>
<ese-Subject>Forests and forestry</ese-Subject>
<ese-Subject>Herbarium</ese-Subject>
<ese-Subject>India</ese-Subject>
<ese-Subject>Indonesia</ese-Subject>
<ese-Subject>Musci</ese-Subject>
<ese-Subject>New Guinea</ese-Subject>
<ese-Subject>Papau New Guinea</ese-Subject>
<ese-Subject>Papua New Guinea</ese-Subject>
<ese-Subject>Plants</ese-Subject>
<ese-Subject>Wallich, N.</ese-Subject>
<ese-Publisher>excudebant Palmblad et c., Upsaliae</ese-Publisher>
<ese-Issued>1825.</ese-Issued>
= <ese-Issued encoding="dmEras">
dmEras:/19th century CE
<thesaurus_link level="1" id="dmEras:50039000" />
</ese-Issued>
<ese-Type>Text</ese-Type>
<ese-Identifier>CallNumber: QK91.C Add 830ab 1825</ese-Identifier>
<ese-Identifier>Abbreviation Musci Buitenzorg</ese-Identifier>
<ese-Source>BHL US</ese-Source>
<ese-HasVersion navigate="BHLUS:ITEMS/000000000007">Examen classis dioeciae. Volume
v. 1 [series pt. 5]</ese-HasVersion>
<ese-HasVersion navigate="BHLUS:ITEMS/000000021856">Examen classis dioeciae.</ese-
HasVersion>
<ese-HasVersion navigate="BHLUS:ITEMS/000000000008">Examen classis dioeciae. Volume
v. 2 [series pt. 5]</ese-HasVersion>
<ese-HasVersion navigate="BHLUS:ITEMS/000000000009">Examen classis dioeciae. Volume
v. 3 [series pt. 5]</ese-HasVersion>
<ese-HasVersion navigate="BHLUS:ITEMS/000000000010">Examen classis dioeciae. Volume
v. 4 [series pt. 5]</ese-HasVersion>
<ese-IsShownAt>http://www.biodiversitylibrary.org/bibliography/2</ese-IsShownAt>
</section>
= <section name="oaiInfo">
<oai-Archive>BHLUS</oai-Archive>
<oai-Set>BHLUS</oai-Set>
<oai-Set>BHLUS:TITLES</oai-Set>
<oai-InternalId>BHLUS:TITLES/000000000002</oai-InternalId>
<oai-DateStamp>2010-09-09T10:09:19Z</oai-DateStamp>
</section>
</regnet-document>

```


III.V Technote: LOCKSS

Koch W.: "Proof of concept LOCKSS (1.Analysis)", 2010-06-30

ECP-2008-DILI-518001

BHL-Europe

Technical Note

Proof of concept LOCKSS (1.Analysis)

Deliverable number	<i>TN-SPRINT1-106</i>
Dissemination level	<i>Public</i>
Delivery date	<i>2010-06-30</i>
Status	<i>Final</i>
Author(s)	<i>Walter Koch</i>



eContentplus

This project is funded under the *eContentplus* programme⁴⁴,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

⁴⁴ OJ L 79, 24.3.2005, p. 1.

Table of contents

1	DOCUMENT HISTORY	108
1.1	CONTRIBUTORS	108
1.2	REVISION HISTORY	108
1.3	DISTRIBUTION.....	108
2	PURPOSE OF THIS DOCUMENT.....	109
3	BACKGROUND AND CONTEXT.....	109
3.1	THE BHL-EUROPE-REQUIREMENT.....	109
3.2	LOCKSS - LOTS OF COPIES KEEP STUFF SAFE.....	110
4	PROPOSED APPROACH.....	110
4.1	LOCKSS AND THE BHL - ARCHITECTURE	110
5	ORGANISATIONAL AND TECHNICAL CONSIDERATIONS	111
5.1	TECHNICAL ISSUES	111
5.1.1	<i>The LOCKSS-Box</i>	112
5.2	ORGANISATIONAL ISSUES	112
6	PROPOSED ACTIONS	113
6.1	CLARIFICATIONS NEEDED	113
6.2	ACTIONS	113

1 Document History

1.1 Contributors

A discussion about this issue was initiated and the following persons provided input that was used for the present document.

Person	Partner
Walter Koch	AIT

1.2 Revision History

Revision Date	Author	Version	Change Reference & Summary
2010-07-15	Walter Koch	0.1	1. Draft
2011-05-03	Chris Sleep	Final	Dissemination changed to public for D3.7

1.3 Distribution

This document has been distributed to:

Group	Date of issue	Version
BHL-Europe – technical group	2010-07-16	0.1

2 Purpose of this document

This document describes the first findings about a possible integration of the LOCKSS-System (Lots of Copies Keep Stuff Safe) into the BHL-Europe System. It should provide a basis for discussion whether it is useful to look deeper into this issue or stop the investigation in this direction in favour of another solution, eg. Bittorrent.

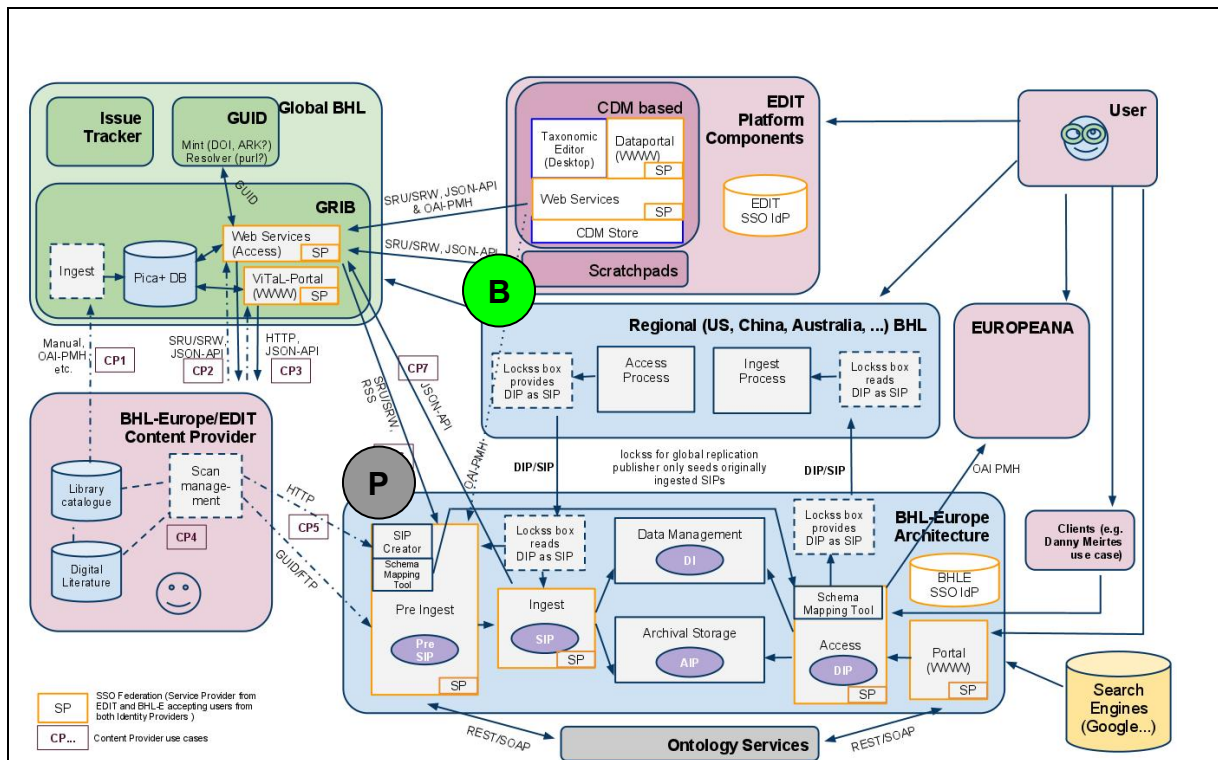
3 Background and Context

3.1 The BHL-Europe -Requirement

The Global BHL System is based on the replication of the large amount of static data (whole BHL data sets) in different regions (America/US, Europe/EU, Australia, Asia/China). There are different methods to achieve this, eg using Bittorrent or LOCKSS-Boxes.

The high level architectural design as discussed in several technical meeting looks like this: (https://bhl.wikispaces.com/BHL-E_WP3_ArchitectureDiagrams - visited: 2010-07-15)

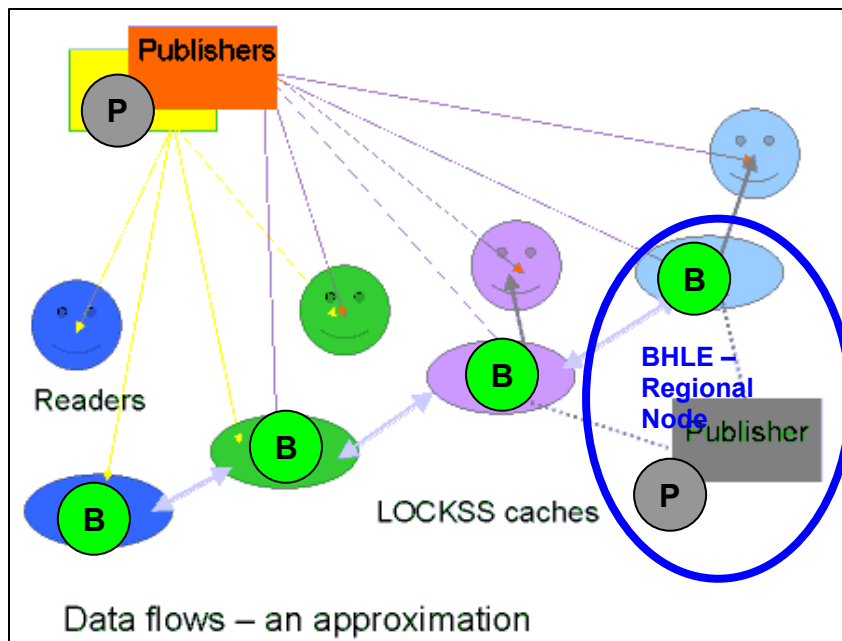
The replication takes place in the middle light blue box (green cycle “B”) in the diagram below. This component could be considered as being the aggregation platform which contains data from all regions. Data belonging only to a region is included in the light blue block at the bottom (grey cycle “P”).



3.2 LOCKSS - Lots of Copies Keep Stuff Safe

In the library world for several years a system for Distributed Digital Preservation has been successfully developed, the LOCKSS System. The system “[...] provides an OAIS-compliant, open source, peer-to-peer, decentralized digital preservation infrastructure. It is format-agnostic, preserving all formats and genres of web-published content, provided the content has an authoritative version.” (http://lockss.org/lockss/How_It_Works#Providing_Access ; URL visited: 2010-07-15).

In an earlier publication of Vicky Reich (2001) the Data Flow in a LOCKSS based distributed system looks like this (figure 1):



“Figure 1: In this example, each LOCKSS cache (oval) collects journal content from the publisher's web site as it is published. Readers (circles) can get content from the publisher site. When the publisher's web site is not available (gray) to a local community, readers from that community get content from their local institution's cache. The caches "talk" to each other to maintain the content's integrity over time.”

(<http://www.dlib.org/dlib/june01/reich/06reich.html> ; URL visited: 2010-07-15).

4 Proposed approach

Locking into the concepts behind LOCKSS and possible implication to the BHL-Europe Project is only one option but will be further pursued in this document. All statements are made on a first conceptual basis and have to be discussed in the appropriate BHL-Europe groupings.

4.1 LOCKSS and the BHL - Architecture

Looking into paragraphs 3.1 and 3.2 it is assumed that on the “regional level” BHL-Europe provides a regional (European) platform which aggregates content from European countries and contributes this content to the global BHL System. This regional system will furtheron called a “BHL-node”. A (the European) BHL-node consists of the two components labeled in par. 3.1 with “P” (to be delivered by the EU-funded project) and “B” respectively. This node interacts with the contributing archives (BHL-Europe -content providers) as well as with “central components” like an ontology node, a “deduplication system” (GRIB), etc.

Aligning this situation with the LOCKSS concept the BHL-Europe -component “P” could be considered as a “publisher” in the LOCKSS environment and the component covering the global BHL data set as a “LOCKSS-Box” containing the cached data from all other Boxes. A “publisher-component” in this concept is responsible for all data generated on a regional level and to be subscribed by the BHL-LOCKSS-Boxes and may be connected to a local production and control system (eg: MetaArchive, PeDALS, or similar implementations of a “PLN” – Private LOCKSS Network).

5 Organisational and Technical Considerations

Implementing a Private LOCKSS network one has to look into organisational and technical requirements and constraints.

5.1 Technical Issues

The whole LOCKSS Software is available in the open source, the software licence is:

Copyright (c) 2000 Board of Trustees of Leland Stanford Jr. University, All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL STANFORD UNIVERSITY BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Stanford University shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from Stanford University.

The software can be downloaded from: <http://sourceforge.net/projects/lockss/> (URL visited: 2010-07-16. Note: the CD image was not the actual one used at that time). The source code is available from: <http://lockss.cvs.sourceforge.net/lockss/> and contains four components:

- The cache manager
- The lockss daemon
- The lockss platform
- The lockss-ui

The whole software was downloaded as GNU tarball and installed on virtual Ubuntu 10.04 box. There have been only minor problems compiling the whole stuff in this environment.

5.1.1 The LOCKSS-Box

The ISO-image which includes the platform and daemon has been downloaded from <http://www.lockss.org/release/lockssCD280.iso> (URL visited: 2010-07-16) and installed as virtual image on a Ubuntu-host machine. The installation needs no special expertise, is straight forward and can be done within half an hour. A test box is now available and running at: <http://lockss.ait.co.at:8081/> (user/password: lockss/lockss).

5.2 Organisational Issues

There are some requirements for running a PLN:

- Members of a PLN have to be member of the LOCKSS-Alliance (annual fee up to 10.800 US\$ per member, depends on organisation type)

Support for PLN members

All members of a PLN must be LOCKSS Alliance members and group discount rates are available. Support includes consulting on hardware and software for LOCKSS boxes, site design and plugin design, implementation and testing, server hosting of configuration parameters, plugins, and title database, etc., and assistance with proxy integration or content export.

- There must be a minimum number of partners (six) running a box in the network.

6 Proposed Actions

Before decision is taken to go ahead with this option (running a distributed BHL archive on global level) some points have to be clarified:

6.1 Clarifications needed

1. Can the software be used without being member of the LOCKSS alliance (in case there is no support needed) ?
2. If answer is Yes – is it possible to get support via the German LuKII project (which has to be agreed by the LOCKSS directors)
3. Can the software be used without any modifications (eg using a simple SIP like defined for bagit)
4. Can a co-operation agreement be made with other PLN providers(eg MetaArchive).

6.2 Actions

1. Decide whether it is useful to go ahead with the LOCKSS approach (Technical Meeting in London)
2. If Yes -> Clarify the points outlined in 6.1 (first contact with Prof. Michael Seadle from the LuKII project has already been established). If No -> stop.
3. Setup a working group preparing the implementation of a PLN-like distributed archival network and integration into the overall BHL architecture.

III.VI Technote: Fedora Commons

Namba L.: “BHL-Europe Fedora Commons Reference Documentation”, 2011-04

Composed of excerpts from documentation taken directly from the Fedora Commons website (<http://fedora-commons.org>).

ECP-2008-DILI-518001

BHL-Europe

Technical Note

BHL-Europe Fedora Commons Reference Documentation

Deliverable number	<i>TN-Addendum-Fedora</i>
Dissemination level	<i>Public</i>
Delivery date	<i>2011-04</i>
Status	<i>Final</i>
Author(s)	<i>Lee Namba</i>



eContentplus

This project is funded under the *eContentplus* programme⁴⁵,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

⁴⁵ OJ L 79, 24.3.2005, p. 1.

Table of contents

1	DOCUMENT HISTORY	117
1.1	CONTRIBUTORS	117
1.2	REVISION HISTORY	117
1.3	DISTRIBUTION	117
2	PURPOSE OF THIS DOCUMENT	118
3	INSTALLATION AND CONFIGURATION	118
3.1	INTRODUCTION	118
3.2	RUNNING THE FEDORA SERVER	120
3.3	DATABASE	120
3.4	CONFIGURING THE FEDORA SERVER	121
4	SERVICE ORIENTED ARCHITECTURE	122
4.1	CORE REPOSITORY SERVICE	122
5	FEDORA DIGITAL OBJECT MODEL	123
5.1	THE FEDORA DIGITAL OBJECT	123
5.2	THE FEDORA DIGITAL OBJECT MODEL	123
5.3	DATASTREAMS	124
5.4	DIGITAL OBJECT MODEL - ACCESS PERSPECTIVE	126
5.5	FOUR TYPES OF FEDORA DIGITAL OBJECTS	127
6	REST API	133
6.1	API-A METHODS	134
6.2	API-M METHODS	140
6.3	UTILITY METHODS	154

6 Document History

6.1 Contributors

This document is a reference document and is composed of excerpts from documentation taken directly from the Fedora Commons website (<http://fedora-commons.org>).

Person	Partner
Lee Namba	ATOS

6.2 Revision History

Revision Date	Author	Version	Change Reference & Summary
2011-04	Lee Namba	0.1	1. Draft
2011-05-03	Chris Sleep	Final	Dissemination changed to public for D3.7

6.3 Distribution

This document has been distributed to:

Group	Date of issue	Version

1 Purpose of this document

This document is a reference document and is composed of excerpts from documentation taken directly from the Fedora Commons website (<http://fedora-commons.org>). We have edited the relevant text as it applies to the BHL-Europe system.

It aims to aggregate the essential Fedora Commons documentation for developers and maintainers of the BHL-Europe system.

2 Installation and Configuration

2.1 Introduction

This guide will show you how to install a new Fedora Repository using the installer, or from source code.

2.1.1.1 Prerequisites

Download Fedora 3.4.2

- [Fedora 3.4.2 Installer \(110M\)](#)
- [Fedora 3.4.2 Source Code \(14M\)](#)

Java SE Development Kit (JDK) 6.

Whether installing a binary or source distribution, JDK 6 is required. The JDK should be installed on the machine you intend to use as the Fedora server. It is available from <http://java.sun.com/>. Look [here](#) for more information on installing Java.

Database

Fedora uses a [MySQL](#) relational database to support some of its functions. To configure Fedora to use the database, please see the [Database](#) section below for further instructions.

Application Server

The Fedora Repository installer includes Tomcat 6.0.20 which is used.

Maven

Fedora uses Maven for its build environment. Maven2 is available from <http://maven.apache.org/>.

2.1.1.2 Prepare Environment Variables

The following environment variables must be correctly defined:

JAVA_HOME

This should point to the base directory of your Java installation. For UNIX derivatives, this might be something like `/usr/local/jdk1.6.0_17`.

FEDORA_HOME

This is the directory where Fedora will be installed, for example, `/usr/local/fedora` (UNIX derivatives). Note: This is only required when running the Fedora client command line utilities. The server also requires this information at run time, but can accept it from the following sources (listed in order of preference):

- The `fedora.home` init-param in the Fedora webapp's `web.xml` file (Fedora 3.2+ only). The installer will automatically include the correct path in your `web.xml` at installation time, so if you move your Fedora Home directory later, you will need to also modify this file and restart the webapp container.
- The `fedora.home` system property, configured as appropriate for your web application server of choice.
- The `FEDORA_HOME` environment variable, as available when the web application server starts.

PATH

This must include the Java and Fedora bin directories. For UNIX derivatives, this will be `$FEDORA_HOME/server/bin`, `$FEDORA_HOME/client/bin` and usually `$JAVA_HOME/bin`.

CATALINA_HOME

Fedora is configured to use Tomcat and `CATALINA_HOME` must be set before starting Fedora. `CATALINA_HOME` should be set to `$FEDORA_HOME/tomcat`.

DISPLAY

(Unix-only)

When running a Fedora server in a Unix-like operating system (Linux, Solaris, OS X, etc), you should ensure that this environment variable is NOT set by the user who will be running the application server in which Fedora is installed (e.g. Tomcat). *Background:* Fedora and the included web applications are designed to run without access to a graphics output device. Although rare, having this environment variable set has been [reported to cause stability problems in certain installations](#) of Fedora. Since a graphic output device should never be needed by the Fedora server, it is safest to ensure this environment variable is not set.

The Fedora Installer provides three installation options: *quick*, *custom*, and *client*. The custom installer is used.

To start the installer, change to the directory where you downloaded the installer and at a command prompt, enter:

```
java -jar fcrepo-installer-3.4.2.jar
```

Please ensure that the user account that is running the installer has sufficient permissions to write to the directories where Fedora will be installed (if deploying to an existing Tomcat installation, this includes permissions to the Tomcat directory). Installer created files will usually be owned by the user running the installer. Consequently, for example, after installation users of the Fedora Admin client will need write permissions to the log files defined by `FEDORA_HOME/client/log4j.xml`.

2.1.1.3 Custom Install

The custom option provides the most flexibility in configuring an installation. Options include the choice of servlet container, database, the host, ports and application server context Fedora will be running on, enabling optional services, as well as security options including SSL, XACML policy enforcement, and FeSL.

Servlet Container

The installer will automatically configure and deploy to Tomcat 5.0.x, 5.5.x, and 6.0.x servlet containers. However, if an existing Tomcat installation (as opposed to the Tomcat bundled with the installer) was selected, the installer will not overwrite your existing `server.xml`, but

rather, place a modified copy at `FEDORA_HOME/install` so that you may review it before installing it yourself.

Other servlet containers will require manual deployment of the *war* files located at `FEDORA_HOME/install`.

Application Server Context

The installer provides the option to enter an application server context name under which Fedora will be deployed. The context name defaults to `Fedora` (resulting in `http[s]://host:port/fedora`), however any other valid context name can be supplied. The installer will name the resulting *war* file according to the supplied context name (defaults to `fedora.war`). Please ensure that the servlet container configuration reflects the name of the Fedora context name in case it needs to be configured explicitly. For further details see [Alternative Webapp Context Configuration](#).

SSL

Configuring SSL support for Fedora's API-M interface is an optional feature. It is strongly recommended for production environments if Fedora is exposed to unsecured application and users. However, since the BHL-Europe installation is within a managed data center with firewall services, SSL will be provided with a reverse proxy implemented using the [Apache HTTP Server](#) thus hiding Fedora and providing better SSL performance.

If the Tomcat servlet container is selected, the installer will configure `server.xml` for you. However, as noted above, if an existing Tomcat installation was selected, the installer will not overwrite your existing `server.xml`.

FeSL

Not used.

Resource Index

Not used.

Messaging

Not used.

2.1.1.4 Client Install

Both the quick and custom options will install the Fedora client software in addition to the Fedora server. The client option, however, will install only the Fedora client software.

2.2 Running the Fedora Server

You will find Tomcat installed in `FEDORA_HOME/tomcat`. To run Fedora, start Tomcat by entering:

```
$FEDORA_HOME/tomcat/bin/startup.sh
```

If you selected the custom install option, ensure that your database server is running.

2.3 Database

Fedora is designed to be RDBMS-independent. Fedora has been tested with Derby, McKoi, MySQL, Oracle, PostgreSQL and Microsoft SQL Server. The embedded version of Derby included with the installer is provided as a convenience; Derby is not recommended for use in production repositories. If you choose to use any database other than the embedded Derby provided by the Fedora Installer, you must install that database first.

Follow the instructions below for MySQL in order to create the user and tables required by Fedora.

2.3.1.1 MySQL

Please note that the MySQL JDBC driver provided by the installer requires MySQL v3.23.x or higher.

The MySQL commands listed below can be run within the `mysql` program, which may be invoked as follows:

```
mysql -u root -p
```

Create the database. For example, to create a database named "fedora3", enter:

```
CREATE DATABASE fedora3;
```

Set username, password and permissions for the database. For example, to set the permissions for user `fedoraAdmin` with password `fedoraAdmin` on database "fedora3", enter:

```
GRANT ALL ON fedora3.* TO fedoraAdmin@localhost IDENTIFIED BY 'fedoraAdmin';
```

```
GRANT ALL ON fedora3.* TO fedoraAdmin@'%' IDENTIFIED BY 'fedoraAdmin';
```

MySQL 4.1.x users must also specify the default character set for the Fedora database as "utf8" and the default collation as "utf8_bin". For example, to set the default character set and collation on a database named "fedora3", enter:

```
ALTER DATABASE fedora3 DEFAULT CHARACTER SET utf8;
```

```
ALTER DATABASE fedora3 DEFAULT COLLATE utf8_bin;
```

2.4 *Configuring the Fedora Server*

2.4.1.1 `fedora.fcfg`

The Fedora Server's configuration is chiefly governed by the Fedora Server Configuration File, `fedora.fcfg`, located at `FEDORA_HOME/server/config/fedora.fcfg`.

The Fedora server configuration file contains:

- Global parameters for the Fedora server
- Configuration parameters for each server module
- Configuration parameters for each persistent data store

The configuration file has a simple schema. It starts with a server element, under which a series of parameter elements occur, followed by a series of module elements, followed by a series of `datastore` elements. The parameter elements directly following the root server element are used to control what are considered generic server functionality; for example: the port on which the server is exposed.

The module elements are used to configure specific parts of Fedora. For instance, the module with the role attribute `fedora.server.search.FieldSearch` is used to configure the field-searching component of the server. Inside the module element, several `param` elements are included. These are specific to that module's implementation. Descriptions of each parameter can currently be found in the configuration file itself.

The `datastore` elements are used to configure various databases that might be used by the system. Although the sample configuration file holds several, you will typically only need one. The `datastore` elements are associated with the modules by means of a parameter inside the associated module. In the sample configuration file, for example, the `poolNames` parameter of the `fedora.server.storage.ConnectionPoolManager` module refers to one of the `datastore` elements in its value.

There are many other parameters you can configure with Fedora. Refer to the Fedora Server Configuration File itself (`fedora.fcfcg`) for internal documentation on all the parameters.

2.4.1.2 Logging in Fedora

Fedora uses the Simple Logging Facade for Java (SLF4J) framework for logging with Logback as the actual logging implementation. For detailed information about using SLF4J, consult the SLF4J Manual: <http://www.slf4j.org/manual.html>, and for information about using Logback consult the Logback manual: <http://logback.qos.ch/manual/index.html>.

The log configuration file is located at `FEDORA_HOME/server/config/logback.xml`. One of the benefits of using SLF4J and Logback is that configuration changes take effect without need to restart the server.

Normally, coarse-grained logs for Fedora are written to `FEDORA_HOME/server/logs/fedora.log`. The following examples show the kinds of configuration changes you can make to aid in debugging.

To change the level to `DEBUG` for all Fedora classes, change the `logger name="org.fcrepo"` line to the following:

```
<logger name="org.fcrepo" additivity="false" level="DEBUG">
```

To change the level to `DEBUG` for just one class, add the following lines:
`log4j.logger.fedora.server.utilities.SQLUtility = DEBUG, FEDORA`

```
log4j.additivity.fedora.server.utilities.SQLUtility = false
```

To change the level to `DEBUG` for a whole package, add the following lines:

```
<logger name="org.fcrepo.server.resourceIndex" additivity="false" level="DEBUG">
```

```
<appender-ref ref="FEDORA"/>
```

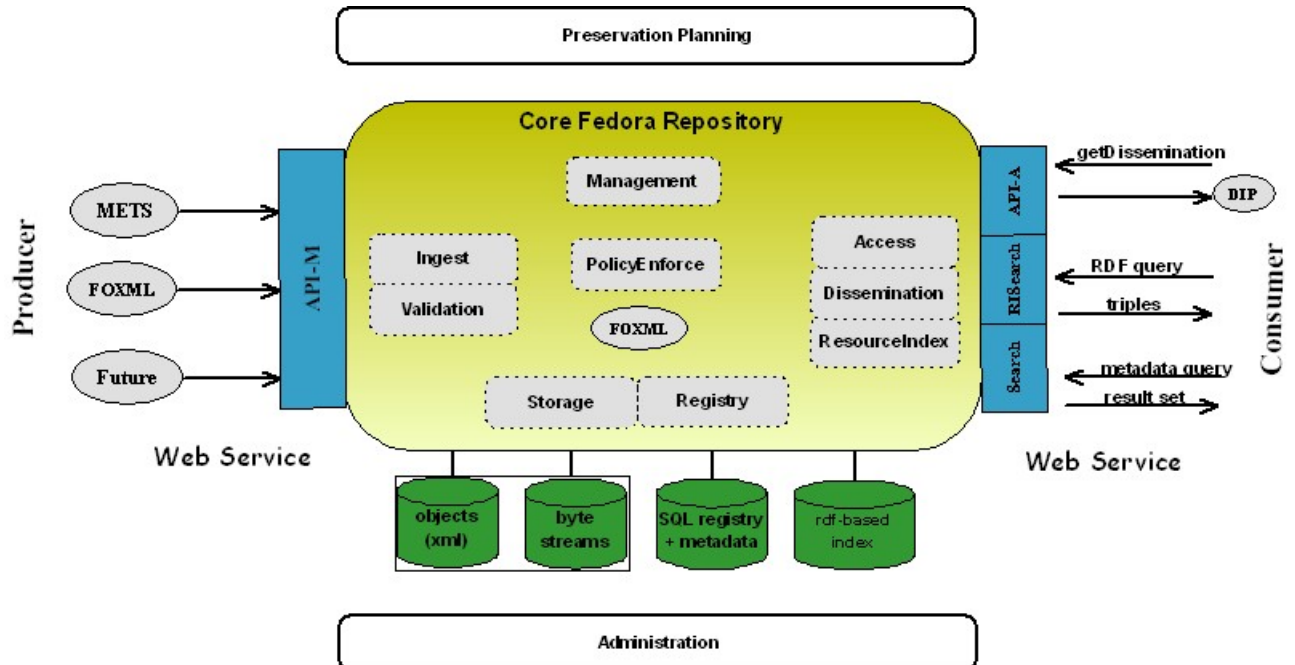
3 Service Oriented Architecture

3.1 Core Repository Service

The Fedora Core Repository Service is run as a stand-alone service. The core repository can be accessed via web service interfaces to its core functionality. The core repository service actually has several web service APIs: an interface for repository management (API-M); an interface for repository access (API-A); interface for basic repository search; and an interface for RDF-based search of the Resource Index. All of these web service interfaces are available on the Fedora repository server web application that runs in Tomcat. The repository service is built in a modular manner, so that each inner function is implemented as a java-based module. The inner modules are configurable, and they can be replaced with alternate implementations.

The Fedora repository service is the core service in the Fedora Service Framework. Below, the Fedora repository service is depicted in more detail, with its inner modules exposed, and

all repository interfaces. The diagram depicts the repository service from the perspective of how it maps to the [Open Archival Information System \(OAIS\)](#) reference model which has been approved as an ISO standard.



4 Fedora Digital Object Model

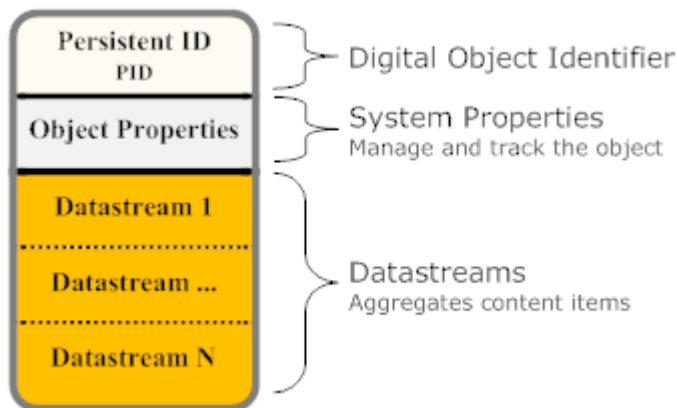
4.1 The Fedora Digital Object

Fedora defines a generic digital object model that can be used to persist and deliver the essential characteristics for many kinds of digital content including documents, images, electronic books, multi-media learning objects, datasets, metadata and many others. This digital object model is a fundamental building block of the Content Model Architecture and all other Fedora-provided functionality.

4.2 The Fedora Digital Object Model

Fedora uses a "compound digital object" design which aggregates one or more content items into the same digital object. Content items can be of any format and can either be stored locally in the repository, or stored externally and just referenced by the digital object. The Fedora digital object model is simple and flexible so that many different kinds of digital objects can be created, yet the generic nature of the Fedora digital object allows all objects to be managed in a consistent manner in a Fedora repository.

A good discussion of the Fedora digital object model (for Fedora 2 and prior versions) exists in a recent paper ([draft](#)) published in the [International Journal of Digital Libraries](#). While some details of this paper have been made obsolete by the CMA (e.g. Disseminators), the core principles of the model are still part of the CMA. The Fedora digital object model is defined in XML schema language (see The [Fedora Object XML](#) - FOXML). For more information, also see the [Introduction to FOXML](#) in the Fedora System Documentation.



The basic components of a Fedora digital object are:

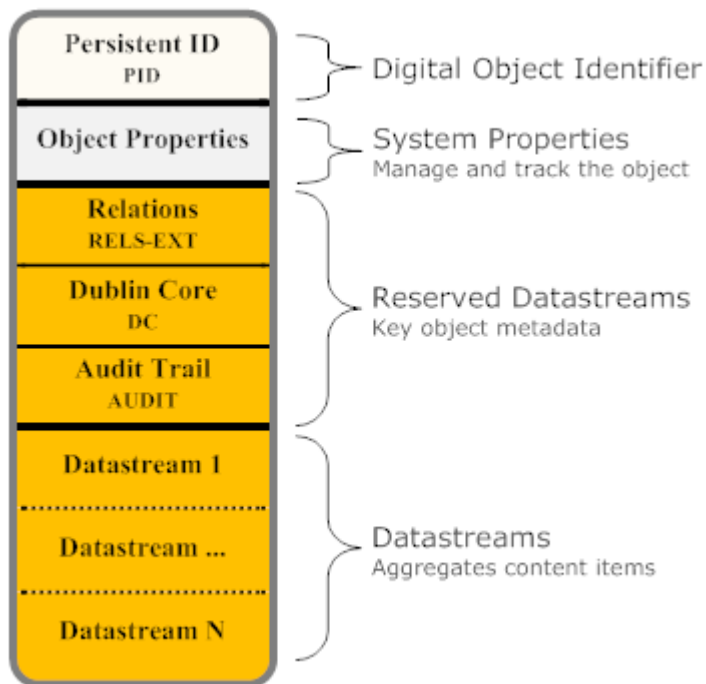
- PID: A persistent, unique identifier for the object.
- Object Properties: A set of system-defined descriptive properties that are necessary to manage and track the object in the repository.
- Datastream(s): The element in a Fedora digital object that represents a content item.

4.3 Datastreams

A Datastream is the element of a Fedora digital object that represents a content item. A Fedora digital object can have one or more Datastreams. Each Datastream records useful attributes about the content it represents such as the MIME-type (for Web compatibility) and, optionally, the URI identifying the content's format (from a format registry). The content represented by a Datastream is treated as an opaque bit stream; it is up to the user to determine how to interpret the content (i.e. data or metadata). The content can either be stored internally in the Fedora repository, or stored remotely (in which case Fedora holds a pointer to the content in the form of a URL). The Fedora digital object model also supports versioning of Datastream content (see the [Fedora Versioning Guide](#) for more information).

Each Datastream is given a Datastream Identifier which is unique within the digital object's scope. Fedora reserves four Datastream Identifiers for its use, "DC", "AUDIT", "RELS-EXT" and "RELS-INT". Every Fedora digital object has one "DC" (Dublin Core) Datastream by default which is used to contain metadata about the object (and will be created automatically if one is not provided). Fedora also maintains a special Datastream, "AUDIT", that records an audit trail of all changes made to the object, and can not be edited since only the system controls it. The "RELS-EXT" Datastream is primarily used to provide a consistent place to describe relationships to other digital objects, and the "RELS-INT" datastream is used to describe internal relationships from digital object datastreams. In addition, a Fedora digital object may contain any number of custom Datastreams to represent user-defined content.

Decisions about what to include in a Fedora digital object and how to configure its Datastreams are choices as you develop content for your repository. The examples in this tutorial demonstrate some common models that you may find useful as you develop your application. Different patterns of datastream designed around particular "genre" of digital object (e.g., article, book, dataset, museum image, learning object) are known as "content models" in Fedora.



The basic properties that the Fedora object model defines for a Datastream are as follows:

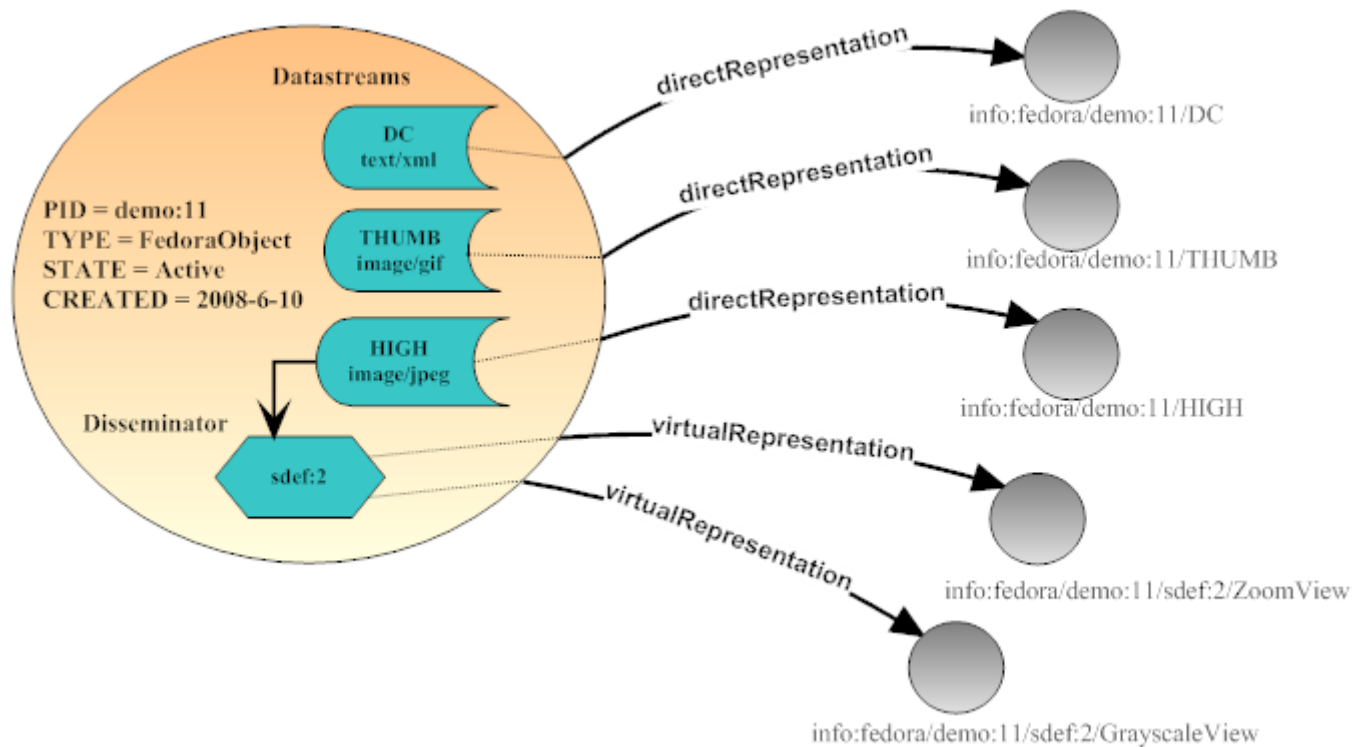
- **Datastream Identifier:** an identifier for the datastream that is unique within the digital object (but not necessarily globally unique)
- **State:** the Datastream's state: Active, Inactive, or Deleted
- **Created Date:** the date/time that the Datastream was created (assigned by the repository service)
- **Modified Date:** the date/time that the Datastream was modified (assigned by the repository service)
- **Versionable:** an indicator (true/false) as to whether the repository service should version the Datastream (by default the repository versions all Datastreams)
- **Label:** a descriptive label for the Datastream
- **MIME Type:** the MIME type of the Datastream (required)
- **Format Identifier:** an optional format identifier for the Datastream such as emerging schemes like PRONOM and the Global Digital Format Registry (GDRF)
- **Alternate Identifiers:** one or more alternate identifiers for the Datastream (such identifiers could be local identifiers or global identifiers such as Handles or DOI)
- **Checksum:** an integrity stamp for the Datastream content which can be calculated using one of many standard algorithms (MD5, SHA-1, etc.)
- **Bytestream Content:** the content (as a stream resource) represented or encapsulated by the Datastream (such as a document, digital image, video, metadata record)
- **Control Group:** the approach used by the Datastream to represent or encapsulate the content as one of four types or control groups:
 - **Internal XML Content** - the content is stored as XML in-line within the digital object XML file
 - **Managed Content** - the content is stored in the repository and the digital object XML maintains an internal identifier that can be used to retrieve the content from storage

- ***Externally Referenced Content*** - the content is stored outside the repository and the digital object XML maintains a URL that can be dereferenced by the repository to retrieve the content from a remote location. While the datastream content is stored outside of the Fedora repository, at runtime, when an access request for this type of datastream is made, the Fedora repository will use this URL to get the content from its remote location, and the Fedora repository will mediate access to the content. This means that behind the scenes, Fedora will grab the content and stream in out the the client requesting the content as if it were served up directly by Fedora. This is a good way to create digital objects that point to distributed content, but still have the repository in charge of serving it up.
- ***Redirect Referenced Content*** - the content is stored outside the repository and the digital object XML maintains a URL that is used to redirect the client when an access request is made. The content is not streamed through the repository. This is beneficial when you want a digital object to have a Datastream that is stored and served by some external service, and you want the repository to get out of the way when it comes time to serve the content up. A good example is when you want a Datastream to be content that is stored and served by a streaming media server. In such a case, you would want to pass control to the media server to actually stream the content to a client (e.g., video streaming), rather than have Fedora in the middle re-streaming the content out.

4.4 Digital Object Model - Access Perspective

Below is an alternative view of a Fedora digital object that shows the object from an access perspective. The digital object contains Datastreams and a set of object properties (simplified for depiction) as described above. A set of access points are defined for the object using the methods described below. Each access point is capable of disseminating a "representation" of the digital object. A representation may be considered a defined expression of part or all of the essential characteristics of the content. In many cases, direct dissemination of a bit stream is the only required access method; in most repository products this is the only supported access method. However, Fedora also supports disseminating virtual representations based on the choices of content modelers and presenters using a full range of information and processing resources. The diagram shows all the access points defined for our example object.

For the access perspective, it would be best if the internal structure of digital object is ignored and treated as being encapsulated by its access points. Each access point is identified by a URI that conforms to the [Fedora "info" URI scheme](#). These URIs can be easily converted to the URL syntax for the Fedora REST-based access service (API-A-LITE). It should be noted that Fedora provides a several protocol-based APIs to access digital objects. These protocols can be used both to access the representation and to obtain associated metadata at the same access point.



By default, Fedora creates one access point for each Datastream to use for direct dissemination of its content. The diagram shows how these access points map to the Datastreams. The example object aggregates three Datastreams: a Dublin Core metadata record, a thumbnail image, and a high resolution image. As shown, each Datastream is accessed from a separate URI.

Custom access points are created using the Content Model Architecture by defining control objects as described below. Behind the scenes, custom access points connect to services that are called on by the repository to produce representations of the object. Custom access points are capable of producing both virtual and direct representations (though they are likely to provide slower performance). Content in the Datastreams may be used as input as well as caller-provided parameters. A "virtual representation" is produced at runtime using any resource the service can access in conjunction with content generated in its code. In this example, there is one service that contains two operations, one for producing zoomable images and one for producing grayscale images. These operations both require a jpeg image as input, therefore the Datastream labeled "HIGH" is used by this service. Fedora will generate one access point for each operation defined by the service. The control objects contains enough information so that a Fedora repository can automatically mediate all interactions with the associated service. The Fedora repository uses this information to make appropriate service calls at run time to produce the virtual representation. From a client perspective this is transparent; the client just requests a dissemination from the desired access point.

4.5 Four Types of Fedora Digital Objects

Although every Fedora digital object conforms to the Fedora object model, as described above, there are four distinct types of Fedora digital objects that can be stored in a Fedora repository. The distinction between these four types is fundamental to how the Fedora repository system works. In Fedora, there are objects that store digital content entities, objects

that store service descriptions, objects used to deploy services, and objects used to organize other objects.

4.5.1.1 Data Object

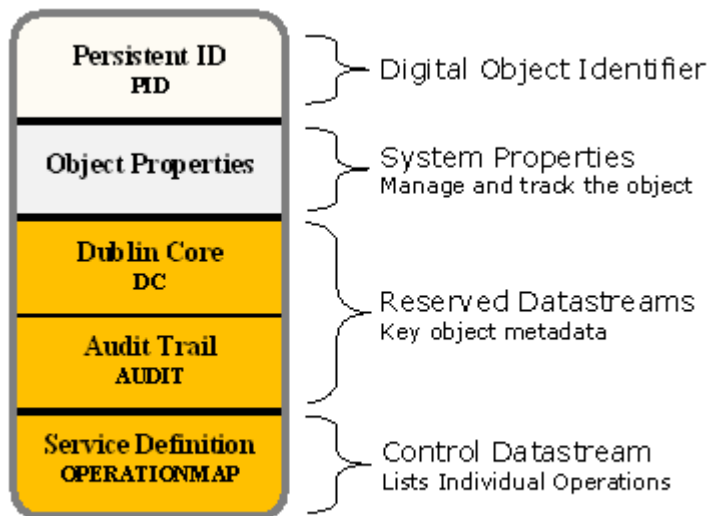
In Fedora, a Data object is the type of object used to represent a digital content entity. Data objects are what we normally think of when we imagine a repository storing digital collections. Data objects can represent such varied entities as images, books, electronic texts, learning objects, publications, datasets, and many other entities. One or more Datastreams are used to represent the parts of the digital content. A Datastream is an XML element that describes the raw content (a bitstream or external content). In the CMA, Disseminators, a metadata construct used to represent services, are eliminated though their functionality is still provided in other ways.

The Data object, indeed all Fedora digital objects, now consists of the FOXML digital object encapsulation (`foxml:digitalObject`) and two fundamental XML elements: Object Properties (`foxml:objectProperties`) and Datastreams (`foxml:datastream`). The Data object is the simplest, most common of all the specialized object types and is identical to the digital object described in the Fedora Digital Object Model section above.

Data objects can now be freely shared between Fedora repositories. If a federated identifier-resolver system, such as the Handle System™, or any authoritative name registry system is used, the Data object will have the same identifier for each copy of itself in each participating repository. Sharing Data objects while keeping the same identifier in each copy greatly simplifies replication, and enables many business processes and services that are needed for large scale repository installations integrated within the Fedora Framework. Data objects can still be shared between repositories by including both the original identifier and alternate identifiers as part of the object's metadata.

4.5.1.2 Service Definition Object

In Fedora, a *Service Definition object* or `SDef` is a special type of *control object* used to store a model of a Service. A Service contains an integrated set of *Operations* that a Data object supports. In object-oriented programming terms, the SDef defines an "interface" which lists the operations that are supported but does not define exactly how each operation is performed. This is also similar to approaches used in Web (REST) programming and in SOAP Web services. In order to execute an operation you need to identify the Data object, the SDef, and the name of the Operation. Some Operations use content from Datastreams (supplied by the Data object) and, possibly, additional parameters supplied by the client program or browser requesting the execution.



Conceptually an Operation is called using the following form (the specifics vary with the actual Fedora interface being used but all will contain some form of this information):

Repository : Get : Data object PID : SDef PID : Operation Name : Optional Parameters

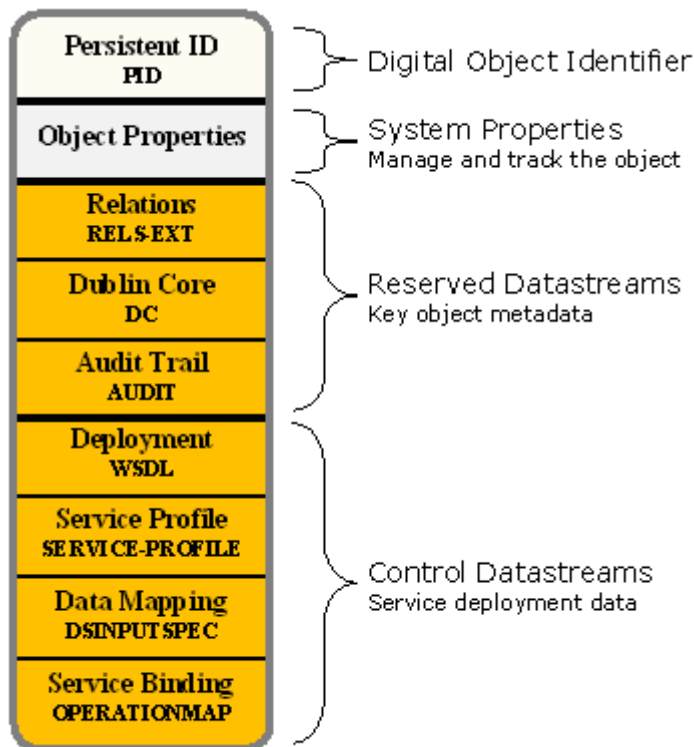
A SDef is a building block in the CMA that enables adding customized functionality for Data objects. Using a SDef is a way of saying "this Data object supports these operations." Essentially, a SDef defines a "behavior contract" to which one or more Data objects may "subscribe." In repositories, we usually create a large number of similar Data objects and want them all to have the same functionality. To make this approach flexible and easier to use, the CMA uses the Content Model (CModel) object (described below) to contain the model for similar Data objects. Instead of associating the SDef directly with each Data object, the relation *hasService* is asserted to the CModel object. By following the relation between the Data object to the CModel object, and then from the CModel object to the SDef object, we can determine what Operations the Data object can perform. Also note that a Data object (through its CModel object) may support more than one Service (by having multiple SDef relations).

SDef objects can now be freely shared between Fedora repositories. If a federated identifier-resolver system, such as the Handle System™, or any authoritative name registry system is used the SDef object will have the same identifier for each copy of itself in each participating repository. Sharing SDef objects while keeping the same identifier in each copy greatly simplifies replication, and enables many business processes and services that are needed for large scale repository installations integrated within the Fedora Framework. SDef objects can still be shared between repositories by including both the original identifier and alternate identifiers as part of the object's metadata. The best results will be gained by sharing the Data object, SDef objects, and Content Model object as a group maintaining the same original identifiers. By using the CMA in this fashion, you transfer a significant unit of the data and metadata that documents the expression pattern for your intellectual work. While this is, by itself, not everything needed, it is a big step forward for creating a durable content repository.

It is worth noting that Service Definition objects conform to the basic Fedora object model. Also, they are stored in a Fedora repository just like other Fedora objects. As such, a collection of SDef objects in a repository constitutes a "registry" of Service Definitions.

4.5.1.3 Service Deployment Object

The Service Deployment object is a special type of control object that describes how a specific repository will deliver the Service Operations described in a SDef for a class of Data objects described in a CModel. The SDep is executable code but instead it contains information that tells the Fedora repository how and where to execute the function that the SDep represents. In the CMA, the SDep acts as a deployment object only for the specific repository in which it is ingested; each repository is free to provide functionality in a different way. For example, one Fedora repository may choose to use a Servlet and another may use a SOAP Web service to perform the same function. As another example, individual repository implementations may need to provide the functionality at different end points. Or perhaps, a specific installation may use a dynamic end point resolution mechanism to permit failover to different service providers.



Since the SDep operates only within the scope of an individual repository, the operators of that repository are free to make changes to the SDep or the functionality it represents at any time (except for temporarily making the object's services unavailable while the change is being made). This approach permits the system operators to control access to services called by the Fedora repository to institute security or policies as their organization determines. It enables Fedora-called services to be managed using the same principles and tools for the deployment of any distributed system. It also enables the system operators to reconfigure their systems quickly without having to change any part of their content except the SDep object.

The SDep stores concrete service binding metadata. A SDep uses a *isDeploymentOf* relation to a SDef as its way of saying "I am able to perform the service methods described by that SDef." A SDep object is related to a SDef in the sense that it defines a particular concrete implementation of the abstract operations defined in a SDef object. The SDef also uses a

isContractorOf relation to a CModel as a way of saying "Use me to do the service operations for any Data objects conforming to that CModel."

A SDep object stores several forms of metadata that describe the runtime bindings for invoking service methods. The most significant of these metadata formats is service binding information encoded in the Web Services Description Language (WSDL). The Fedora repository system uses the WSDL at runtime to dispatch service method requests in fulfilling client requests for "virtual representations" of a Data object (i.e., via its Operations). This enables Fedora to talk to a variety of different services in a predictable and standard manner. A SDep also contains metadata that defines a "data contract" between the service and a class of Fedora Data objects as defined in the CModel. For the initial deployment of the CMA a simple data contract mechanism was chosen. Since the Datastream IDs are specified in the CModel and the SDep is now a deployment control object only for a specific repository, the SDep is able to uniformly bind directly to these IDs. In the future a more abstract binding mechanism may be used but this approach is simple and clear, though it may require the creation of a small number of additional SDep objects.

A major aspect of the CMA redesign is that there is no requirement that conformance to a Content Model or that referential integrity between objects be checked at ingest time. This may result in a run-time error if the repository cannot find referenced objects, interpret the Content Model or if there are any conformance problems.

It is worth noting that SDep objects conform to the basic Fedora object model. Also, they are stored in a Fedora repository just like other Fedora objects. As such, a collection of SDep objects in a repository constitutes a "registry" of service deployments that can be used with Fedora objects. In the CMA, SDep objects are not freely sharable across repositories. They represent how a specific repository implements a service. However, SDep objects can be shared if the operator of the system modifies them for local deployment. Because of this, SDep objects should not be automatically replicated between repositories without considering the affect.

4.5.1.4 Content Model Object

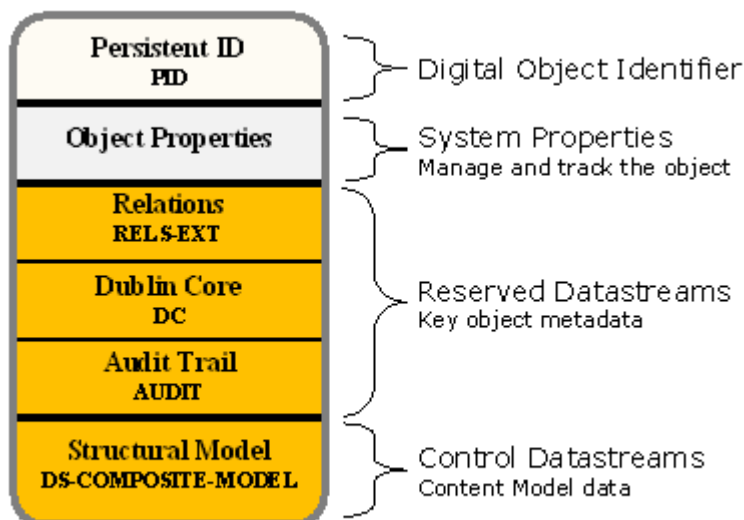
The Content Model object or CModel is a new specialized control object introduced as part of the CMA. It acts as a container for the Content Model document which is a formal model that characterizes a class of digital objects. It can also provide a model of the relationships which are permitted, excluded, or required between groups of digital objects. All digital objects in Fedora including Data, SDef, SDep, and CModel objects are organized into classes by the CModel object. In this section, we will primarily discuss the relationship between the Data and CModel objects.

To create a class of Data objects, create a CModel object. Each Data object belonging to the class asserts the relation *hasModel* using the identifier of the CModel as the object of the assertion. The current CModel object contains a structural model of the Data object. Over time there will be additional elements to the Content Model document but this initial implementation is sufficient to describe the Datastreams which are required to be present in each Data object in the class. The other key relation is to the SDef objects. You can assert zero or more *hasService* relations in the CModel to SDef objects.

A Data object may assert a *hasModel* relationship to multiple CModel objects. Such a Data object should conform to all of its Content Models, containing an aggregation of all the Datastreams defined by the Content Models. If two or more Content Models define Datastreams which have the same name but different characteristics, no well-formed Data

object can be constructed and likely the repository will be unable to deliver its content or services. Fedora automatically assumes that all objects conform to a system-defined "Basic Content Model." There is no need to assert a relation to this content model explicitly but, if the Data object asserts other relations, it is a good practice to make the assertion to the Basic Content Model explicit. Regardless, the repository will behave the same whether the relation is asserted or not. Along with the Basic Content Model, the repository defines a "Basic Service Definition" which supplies Operations common to all objects. One such service provides direct access to the Datastreams.

Because of the Basic Content Model and the Basic Service Definition, nothing needs to be added to a Data object if the user only wants to store and disseminate Datastreams by name. However, without an explicit Content Model you cannot validate whether the Data object is correctly formed. In the CMA, if the repository cannot find and interpret all the control objects related to a Data object, or cannot interpret the Content Model, it will issue a runtime error when the Data object is accessed. Note that the repository will always be able to perform basic Datastream operations because they are a part of the Basic Content Model and Basic Service Definition. Other than conformance to the rules for a properly formed digital object, there is no warning or error issued on ingest or modification of an object in the CMA.



CModel objects can now be freely shared between Fedora repositories. If a federated identifier-resolver system, such as the Handle System™, or any authoritative name registry system is used the CModel object will have the same identifier for each copy of itself in each participating repository. Sharing CModel objects while keeping the same identifier in each copy greatly simplifies replication, and enables many business processes and services that are needed for large scale repository installations integrated within the Fedora Framework. CModel objects can still be shared between repositories by including both the original identifier and alternate identifiers as part of the object's metadata. The best results will be gained by sharing the Data object, SDef objects, and CModel objects as a group maintaining the same original identifiers. By using the CMA in this fashion, you transfer a significant unit of the data and metadata that documents the expression pattern for your intellectual work. While this is, by itself, not everything needed, it is a big step forward for creating a durable content repository. Over time, Content Model languages can be developed that permit

describing an ever larger portion of the essential characteristics of the content and its behaviors.

It is worth noting that Content Model objects conform to the basic Fedora object model. Also, they are stored in a Fedora repository just like other Fedora objects. As such, a collection of Content Model objects in a repository constitutes a "registry" of Content Models.

5 REST API

The Fedora REST API exposes a subset of the Fedora Access and Management APIs as a RESTful (Representational State Transfer) Web Service.

For examples of how to use the REST API programmatically, please refer to the [TestRESTAPI test class](#).

- ⊖ Ensure DC, RELS-EXT and RELS-INT are versionable if using Managed Content**
Due to an outstanding bug [FCREPO-849](#), if you use Managed Content for DC, RELS-EXT or RELS-INT then please make sure these datastreams are versionable (the default setting for versionable is "true", so if you haven't specified this datastream property then you are safe). Ensure that you don't inadvertently set this property to "false" for these datastreams when using the API methods.
- ⓘ 2xx Responses only please**
The HTTP Response portion of each method description listed below indicates the response on success. Unsuccessful calls will produce non-200 response codes appropriate to the error case. If, however, your client software has difficulty processing non-200 responses (such as is the case with Adobe's Flash Player) adding the query parameter 'flash=true' to any method will ensure that all responses are in the 200 range. In the event of an error, the response code will be set to 200 and the response body will include the error message followed by "::ERROR".
- ⓘ POST Replacement**
If the client with which you are working does not support use of the PUT and/or DELETE HTTP methods but does allow you to set headers on the HTTP request, you can use POST replacement to make PUT and DELETE calls. To do this, simply set the X-HTTP-Method-Override request header to the correct method value (PUT or DELETE) and perform a POST request. Your request will be handled by the REST API as if it were a PUT or DELETE.
- ⓘ Removal of .xml shortcut**
For release 3.3 the `.xml` shortcut has entirely been removed from the REST API due to functional inconsistencies (see [here](#) for more details). If your client uses this shortcut please change it to use the format parameter (`?format=xml`).
- ⓘ URL-Encoding**
The REST API requires that parameters - including path parameters - are URL-encoded. Particularly this is important if you have any PIDs that use [escaped-octets](#) in the PID name. In this case the "%" character should be URL-encoded as "%25", eg a PID "changeme:1234%2F56" should be URL-encoded as "changeme:1234%252F56". The ":" PID namespace separator character does not require URL-encoding as it has no special meaning when used in the path component of HTTP URIs; however some software library URL-encoding methods will URL-encode this to %3A - that's not a problem, both ":" and "%3A" will be accepted by the REST API.

5.1 API-A Methods

5.1.1.1 describeRepository

Not implemented

5.1.1.2 findObjects

URL Syntax

/objects ? [terms | query] [maxResults] [resultFormat] [pid] [label] [state] [ownerId] [cDate] [mDate] [dcmDate] [title] [creator] [subject] [description] [publisher] [contributor] [date] [type] [format] [identifier] [source] [language] [relation] [coverage] [rights]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
terms	a phrase represented as a sequence of characters (including the ? and * wildcards) for the search. If this sequence is found in any of the fields for an object, the object is considered a match. Do NOT use this parameter in combination with the "query" parameter		
query	a sequence of space-separated conditions. A condition consists of a metadata element name followed directly by an operator, followed directly by a value. Valid element names are (pid, label, state, ownerId, cDate, mDate, dcmDate, title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, rights). Valid operators are: contains (), equals (=), greater than (>), less than (<), greater than or equals (>=), less than or equals (<=). The contains () operator may be used in combination with the ? and * wildcards to query for simple string patterns. Space-separators should be encoded in the URL as %20. Operators must be encoded when used in the URL syntax as follows: the (=) operator must be encoded as %3D, the (>) operator as %3E, the (<) operator as %3C, the (>=) operator as %3E%3D, the (<=) operator as %3C%3D, and the (~) operator as %7E. Values may be any string. If the string contains a space, the value should begin and end with a single quote character ('). If all conditions are met for an object, the object is considered a match. Do NOT use this parameter in combination with the "terms" parameter		
maxResults	the maximum number of results that the server should provide at once. If this is unspecified, the server will	25	

	default to a small value		
resultFormat	the preferred output format	html	xml, html
pid	if true, the Fedora persistent identifier (PID) element of matching objects will be included in the response	false	true, false
label	if true, the Fedora object label element of matching objects will be included in the response	false	true, false
state	if true, the Fedora object state element of matching objects will be included in the response	false	true, false
ownerId	if true, each matching objects' owner id will be included in the response false>true, false	false	true, false
cDate	if true, the Fedora create date element of matching objects will be included in the response	false	true, false
mDate	if true, the Fedora modified date of matching objects will be included in the response	false	true, false
dcmDate	if true, the Dublin Core modified date element(s) of matching objects will be included in the response	false	true, false
title	if true, the Dublin Core title element(s) of matching objects will be included in the response	false	true, false
creator	if true, the Dublin Core creator element(s) of matching objects will be included in the response	false	true, false
subject	if true, the Dublin Core subject element(s) of matching objects will be included in the response	false	true, false
description	if true, the Dublin Core description element(s) of matching objects will be included in the response	false	true, false
publisher	if true, the Dublin Core publisher element(s) of matching objects will be included in the response	false	true, false
contributor	if true, the Dublin Core contributor element(s) of matching objects will be included in the response	false	true, false
date	if true, the Dublin Core date element(s) of matching objects will be included in the response	false	true, false
type	if true, the Dublin Core type element(s) of matching objects will be included in the response	false	true, false
format	if true, the Dublin Core format element(s) of matching objects will be included in the response	false	true, false
identifier	if true, the Dublin Core identifier element(s) of matching objects will be included in the response	false	true, false
source	if true, the Dublin Core source element(s) of matching objects will be included in the response	false	true, false
language	if true, the Dublin Core language element(s) of matching objects will be included in the response	false	true, false
relation	if true, the Dublin Core relation element(s) of matching objects will be included in the response	false	true, false
coverage	if true, the Dublin Core coverage element(s) of matching objects will be included in the response	false	true, false
rights	if true, the Dublin Core rights element(s) of matching objects will be included in the response	false	true, false

Examples

/objects?terms=demo&pid=true&subject=true&label=true&resultFormat=xml

/objects?query=title%7Erome%20creator%7Estaples&pid=true&title=true&creator=true

/objects?query=pid%7E*1&maxResults=50&format=true&pid=true&title=true

5.1.1.3 getDatastreamDissemination

URL Syntax

/objects/{pid}/datastreams/{dsID}/content ? [asOfDateTime] [download]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{dsID}	datastream identifier		
asOfDateTime	indicates that the result should be relative to the digital object as it existed at the given date and time		yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ
download	If true, a content-disposition header value "attachment" will be included in the response, prompting the user to save the datastream as a file. A content-disposition header value "inline" will be used otherwise. The filename used in the header is generated by examining in order: RELS-INT for the relationship fedora-model:downloadFilename, the datastream label, and the datastream ID. The file extension (apart from where the filename is specified in RELS-INT) is determined from the MIMETYPE. The order in which these filename sources are searched, and whether or not to generate an extension from the MIMETYPE, is configured in fedora.fcfig. The file used to map between MIMETYPES and extensions is mime-to-extensions.xml located in the server config directory.		

Examples

/objects/demo:29/datastreams/DC/content

/objects/demo:29/datastreams/DC/content?asOfDateTime=2008-01-01

5.1.1.4 getDissemination

URL Syntax

/objects/{pid}/methods/{sdefPid}/{method} ? [method parameters]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{sdefPid}	persistent identifier of the sDef defining the methods		
{method}	method to invoke		
method parameters	any parameters required by the method		

Examples

/objects/demo:29/methods/demo:27/resizeImage?width=100

/objects/demo:SmileyEarring/methods/demo:DualResolution/fullSize

5.1.1.5 getObjectHistory

URL Syntax

/objects/{pid}/versions ? [format]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
format	the preferred output format	html	xml, html

Examples

/objects/demo:29/versions

/objects/demo:29/versions?format=xml

5.1.1.6 getObjectProfile

URL Syntax

/objects/{pid} ? [format] [asOfDateTime]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
format	the preferred output format	html	xml, html
asOfDateTime	indicates that the result should be relative to the digital object as it existed on the given date		yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ

Examples

/objects/demo:29

/objects/demo:29?format=xml

/objects/demo:29?asOfDateTime=2008-01-01

5.1.1.7 listDatastreams

URL Syntax

/objects/{pid}/datastreams ? [format] [asOfDateTime]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
format	the preferred output format	html	xml, html
asOfDateTime	indicates that the result should be relative to the digital object as it existed on the given date		yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ

Examples

/objects/demo:35/datastreams

/objects/demo:35/datastreams?format=xml&asOfDateTime=2008-01-01T05:15:00Z

5.1.1.8 listMethods

URL Syntax

1. /objects/{pid}/methods ? [format] [asOfDateTime]
2. /objects/{pid}/methods/{sdefPid} ? [format] [asOfDateTime]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{sdefPid}	persistent identifier of the SDef defining the methods		
format	the preferred output format	html	xml, html
asOfDateTime	indicates that the result should be relative to the digital object as it existed on the given date		yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ

Examples

/objects/demo:29/methods

/objects/demo:29/methods?format=xml&asOfDateTime=2008-01-01T05:15:00Z

/objects/demo:29/methods/demo:27

/objects/demo:29/methods/demo:27?format=xml&asOfDateTime=2008-01-01T05:15:00Z

5.1.1.9 resumeFindObject

URL Syntax

/objects ? [sessionToken] [all findObjects options]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
sessionToken	the identifier of the session to which the search results are being returned		
all findObjects options	all of the same options are available for resumeFindObject as for findObjects		

Examples

/objects?terms=*&format=xml&pid=true&subject=true&label=true&sessionToken=xyz\\

5.2 API-M Methods

5.2.1.1 addDatastream

URL Syntax

/objects/{pid}/datastreams/{dsID} ? [controlGroup] [dsLocation] [altIDs] [dsLabel] [versionable] [dsState] [formatURI] [checksumType] [checksum] [mimeType] [logMessage]

HTTP Method

POST

HTTP Response

201

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{dsID}	datastream identifier		
controlGroup	one of "X", "M", "R", or "E" (Inline *X*ML, *M*anaged Content, *R*edirect, or *E*xternal Referenced)	X	X, M, R, E
dsLocation	location of managed or external datastream content		
altIDs	alternate identifiers for the datastream		
dsLabel	the label for the datastream		
versionable	enable versioning of the datastream	true	true, false
dsState	one of "A", "I", "D" (*A*ctive, *I*nactive, *D*eleted)	A	A, I, D
formatURI	the format URI of the datastream		
checksumType	the algorithm used to compute the checksum	DEFAULT	DEFAULT, DISABLED, MD5, SHA-1, SHA-256, SHA-385, SHA-512
checksum	the value of the checksum represented as a hexadecimal string		
mimeType	the MIME type of the content being added, this overrides the Content-Type request header		
logMessage	a message describing the activity being performed		
multipart file as request content	datastream file (for Managed datastreams)		

Examples

POST: /objects/demo:29/datastreams/NEWDS?controlGroup=X&dsLabel=New (with Multipart file)

POST:
/objects/demo:29/datastreams/NEWDS?controlGroup=M&dsLocation=<http://example:80/newds&dsLabel=New>

5.2.1.2 addRelationship

URL Syntax

/objects/{pid}/relationships/new ? [subject] [predicate] [object] [isLiteral] [datatype]

HTTP Method

POST

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
subject	subject of the relationship. Either a URI for the object or one of its datastreams	URI of this object	
predicate	predicate of the relationship		
object	object of the relationship		
isLiteral	true if the object of the relationship is a literal, false if it is a URI		true, false
datatype	if the object is a literal, the datatype of the literal (optional)		

Examples

POST
/objects/demo:29/relationships/new?subject=info%3afedora%2fdemo%3a29%2fDC&predicate=http%3a%2f%2fwww.example.org%2frels%2fname&object=dublin%20core&isLiteral=true

5.2.1.3 compareDatastreamChecksum

See [getDatastream](#)

5.2.1.4 export

URL Syntax

/objects/{pid}/export ? [format] [context] [encoding]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
format	the XML format to export	info:fedora/fedora-system:FOXML-1.1	info:fedora/fedora-system:FOXML-1.1, info:fedora/fedora-system:FOXML-1.0, info:fedora/fedora-system:METSFedoraExt-1.1, info:fedora/fedora-system:METSFedoraExt-1.0, info:fedora/fedora-system:ATOM-1.1, info:fedora/fedora-system:ATOMZip-1.1
context	the export context, which determines how datastream URLs and content are represented	public	public, migrate, archive
encoding	the preferred encoding of the exported XML	UTF-8	

Examples

/objects/demo:29/export

/objects/demo:29/export?context=migrate

5.2.1.5 getDatastream

URL Syntax

/objects/{pid}/datastreams/{dsID} ? [asOfDateTime] [format] [validateChecksum]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{dsID}	datastream identifier		
format	the preferred output format	html	xml, html
asOfDateTime	indicates that the result should be relative to the digital object as it		yyyy-MM-dd or yyyy-MM-

	existed on the given date		ddTHH:mm:ssZ
validateChecksum	verifies that the Datastream content has not changed since the checksum was initially computed. If asOfDateTime is null, Fedora will use the most recent version.	false	true, false

Examples

/objects/demo:29/datastreams/DC

/objects/demo:29/datastreams/DC?format=xml

/objects/demo:29/datastreams/DC?format=xml&validateChecksum=true

5.2.1.6 getDatastreamHistory

URL Syntax

/objects/{pid}/datastreams/{dsid}/versions ? [format]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
format	the preferred output format	html	xml, html

Examples

GET: /objects/changeme:1/datastreams/DC/versions

GET: /objects/changeme:1/datastreams/DC/versions?format=xml

5.2.1.7 getDatastreams

Not implemented

5.2.1.8 getNextPID

URL Syntax

/objects/nextPID ? [numPIDs] [namespace] [format]

HTTP Method

POST

HTTP Response

200

Parameters

Name	Description	Default	Options
numPIDs	the number of pids to retrieve	1	
namespace	the namespace of the requested pid(s)	the default namespace of the repository	

format	the preferred output format	html	xml, html
---------------	-----------------------------	------	--------------

Examples

POST: /objects/nextPID

POST: /objects/nextPID?numPIDs=5&namespace=test&format=xml

5.2.1.9 getObjectXML

URL Syntax

/objects/{pid}/objectXML

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		

Examples

/objects/demo:29/objectXML

5.2.1.10 getRelationships

URL Syntax

/objects/{pid}/relationships ? [subject] [predicate] [format]

HTTP Method

GET

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
subject	subject of the relationship(s). Either a URI for the object or one of its datastreams	URI of this object	
predicate	predicate of the relationship(s), if missing returns all predicates		
format	format of the response	rdf/xml	xml (returns rdf/xml), rdf/xml, n-triples, turtle, sparql

Examples

/objects/demo:29/relationships

/objects/demo:29/relationships?subject=info%3afedora%2fdemo%3a29%2fDC

5.2.1.11 ingest

URL Syntax

/objects/ [{pid}| new] ? [label] [format] [encoding] [namespace] [ownerId] [logMessage] [ignoreMime]

HTTP Method

POST

HTTP Response

201

Request Content

text/xml

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the object to be created	new (see below)	
new	indicator that either a new PID should be created for this object or that the PID to be used is encoded in the XML included as the body of the request		
label	the label of the new object		
format	the XML format of the object to be ingested	info:fedora/fedora-system:FOXML-1.1, info:fedora/fedora-system:FOXML-1.0, info:fedora/fedora-system:METSFedoraExt-1.1, info:fedora/fedora-system:METSFedoraExt-1.0, info:fedora/fedora-system:ATOM-1.1, info:fedora/fedora-system:ATOMZip-1.1	
encoding	the encoding of the XML to be ingested. If this is specified, and given as anything other than UTF-8, you must ensure that the same encoding is declared in the XML. For example, if you specify "ISO-88591" as the encoding, the XML should start with: <?xml version="1.0" encoding="ISO-8859-1"?>	UTF-8	
namespace	the namespace to be used to	the default namespace of the	

	create a PID for a new empty object; if object XML is included with the request, the namespace parameter is ignored	repository	
ownerId	the id of the user to be listed at the object owner		
logMessage	a message describing the activity being performed		
ignoreMime	indicates that the request should not be checked to ensure that the content is XML prior to attempting an ingest. This is provided to allow for client applications which do not indicate the correct Content-Type when submitting a request.	false	true, false
XML file as request content	file to be ingested as a new object		

Notes

Executing this request with no request content will result in the creation of a new, empty object (with either the specified PID or a system-assigned PID). The new object will contain only a minimal DC datastream specifying the dc:identifier of the object.

Examples

POST: /objects/new

POST: /objects

POST: /objects/new?namespace=demo

POST: /objects/test:100?label=Test

5.2.1.12 modifyDatastream

URL Syntax

/objects/{pid}/datastreams/{dsID} ? [dsLocation] [altIDs] [dsLabel] [versionable] [dsState] [formatURI] [checksumType] [checksum] [mimeType] [logMessage] [ignoreContent] [lastModifiedDate]

HTTP Method

PUT

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{dsID}	datastream identifier		
dsLocation	location of datastream content		
altIDs	alternate identifiers for the datastream		
dsLabel	the label for the datastream		
versionable	enable versioning of the datastream	the "versionable" property of the existing datastream	true, false
dsState	one of "A", "I", "D" (*A*ctive, *I*nactive, *D*eleted)	A	A, I, D
formatURI	the format URI of the datastream		
checksumType	the algorithm used to compute the checksum	DEFAULT	DEFAULT, DISABLED, MD5, SHA-1, SHA-256, SHA-385, SHA-512
checksum	the value of the checksum represented as a hexadecimal string		
contentType	the MIME type of the content being added, this overrides the Content-Type request header		
logMessage	a message describing the activity being performed		
ignoreContent	tells the request handler to ignore any content included as part of the request, indicating that you do not intend to update the datastream content. This is primarily provided to allow the use of client tools which always require content to be included as part of PUT requests.	false	true, false
lastModifiedDate	date/time of the last (known) modification to the datastream, if the actual last modified date is later, a 409 response is returned		
multipart file as request content	file to replace existing datastream (for Managed datastreams)		

Examples

PUT: /objects/demo:35/datastreams/HIGH (with Multipart file)

PUT:

/objects/demo:35/datastreams/HIGH?dsLocation=<http://example:80/highDS?logMessage=Update>

5.2.1.13 modifyObject

URL Syntax

/objects/{pid} ? [label] [ownerId] [state] [logMessage] [\lastModifiedDate]

HTTP Method

PUT

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
label	the new object label		
ownerId	the id of the user to be listed at the object owner		
state	the new object state - *A*ctive, *I*nactive, or *D*eleted	A	A, I, D
logMessage	a message describing the activity being performed		
lastModifiedDate	date/time of the last (known) modification to the datastream, if the actual last modified date is later, a 409 response is returned		

Examples

PUT: /objects/demo:29?label=Updated

PUT: /objects/demo:29?state=D?logMessage=Deleted

5.2.1.14 purgeDatastream

URL Syntax

/objects/{pid}/datastreams/{dsID} ? [startDT] [endDT] [logMessage]

HTTP Method

DELETE

HTTP Response

200 with a string array of the date-time stamps of the versions purged

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{dsID}	datastream identifier		
startDT	the (inclusive) start date-time stamp of the		yyyy-MM-dd or

	range. If not specified, this is taken to be the lowest possible value, and thus, the entire version history up to the endDT will be purged		yyyy-MM-ddTHH:mm:ssZ
endDT	the (inclusive) ending date-time stamp of the range. If not specified, this is taken to be the greatest possible value, and thus, the entire version history back to the startDT will be purged		yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ
logMessage	a message describing the activity being performed		

Examples

DELETE: /objects/demo:35/datastreams/HIGH

5.2.1.15 purgeObject

URL Syntax

/objects/{pid} ? [logMessage]

HTTP Method

DELETE

HTTP Response

204

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
logMessage	a message describing the activity being performed		

Examples

DELETE: /objects/demo:29

5.2.1.16 purgeRelationship

URL Syntax

/objects/{pid}/relationships ? [subject] [predicate] [object] [isLiteral] [datatype]

HTTP Method

DELETE

HTTP Response

200

Return body

Text indicating if the relationship was successfully purged: `true` or `false`

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
subject	subject of the relationship. Either a URI for the object or one of its datastreams	URI of this object	
predicate	predicate of the relationship		
object	object of the relationship		
isLiteral	true if the object of the relationship is a literal, false if it is a URI		true, false
datatype	if the object is a literal, the datatype of the literal (optional)		

Examples

DELETE

/objects/demo:29/relationships?subject=info%3afedora%2fdemo%3a29%2fDC&predicate=ht
tp%3a%2f%2fwww.example.org%2frels%2fname&object=dublin%20core&isLiteral=true

5.2.1.17 setDatastreamState

URL Syntax

/objects/{pid}/datastreams/{dsID} ? [dsState]

HTTP Method

PUT

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{dsID}	datastream identifier		
dsState	one of "A", "I", "D" (*A*ctive, *I*nactive, *D*eleted)	A	A, I, D

Examples

PUT: /objects/demo:35/datastreams/HIGH?dsState=D

5.2.1.18 setDatastreamVersionable

URL Syntax

/objects/{pid}/datastreams/{dsID} ? [versionable]

HTTP Method

PUT

HTTP Response

200

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
{dsID}	datastream identifier		
versionable	enable versioning of the datastream	true	true, false

Examples

PUT: /objects/demo:35/datastreams/HIGH?versionable=false

5.2.1.19 Validate

URL Syntax

/objects/{pid}/validate ? [asOfDateTime]

HTTP Method

GET

HTTP Response

200 (OK) if the validation could be carried out (even if the object is not valid)

404 If some object or datastream could not be carried out

401 If the user credentials was insufficient

400 If the parameters are misformed

409 If one of the relevant objects were locked

500 If something else failed on the server

Return Body

XML, adhering to this schema

```
<xs:schema targetNamespace="http://www.fedora.info/definitions/1/0/access/"
  xmlns="http://www.fedora.info/definitions/1/0/access/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="validation">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="asOfDateTime"/>
        <xs:element ref="contentModels"/>
        <xs:element ref="problems"/>
        <xs:element ref="datastreamProblems"/>
      </xs:sequence>
      <xs:attribute name="pid" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<xs:attribute name="valid" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="asOfDateTime">
  <xs:simpleType>
    <xs:restriction base="xs:dateTime"/>
  </xs:simpleType>
</xs:element>
<xs:element name="contentModels">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="model" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="problems">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="problem" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="datastreamProblems">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="datastream" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="datastream">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="problem" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```

</xs:sequence>
<xs:attribute name="datastreamID" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:schema>

```

Parameters

Name	Description	Default	Options
{pid}	persistent identifier of the digital object		
asOfDateTime	indicates that the result should be relative to the digital object and the repository as it existed at the given date and time		yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ

Examples

GET </objects/validate/demo:29?asOfDateTime=2008-01-01>
Returns HTTP 200 with the body

```

<?xml version="1.0" encoding="UTF-8"?>
<validation pid="demo:29" valid="true">
  <asOfDateTime>2008-01-01T00:00:00.000Z</asOfDateTime>
  <contentModels>
    <model>info:fedora/fedora-system:FedoraObject-3.0</model>
  </contentModels>
  <problems>
  </problems>
  <datastreamProblems>
  </datastreamProblems>
</validation>

```

GET </objects/validate/demo:fail>
Returns HTTP 200 with the body. Here the error was a misspelled element in the DC datastream, "dc:titel"

```

<?xml version="1.0" encoding="UTF-8"?>
<validation pid="demo:fail" valid="false">
  <asOfDateTime></asOfDateTime>
  <contentModels>
    <model>info:fedora/fedora-system:FedoraObject-3.0</model>
  </contentModels>

```

```
<problems>
</problems>
<datastreamProblems>
  <datastream datastreamID="DC">
```

```
    <problem>Encountered schema validation error while parsing datastream 'DC' with the schema
from content model 'fedora-system:FedoraObject-3.0'. The error was 'cvc-complex-type.2.4.a: Invalid
content was found starting with element 'dc:title'. One of '{"http://purl.org/dc/elements/1.1/":title,
"http://purl.org/dc/elements/1.1/":creator, "http://purl.org/dc/elements/1.1/":subject,
"http://purl.org/dc/elements/1.1/":description, "http://purl.org/dc/elements/1.1/":publisher,
"http://purl.org/dc/elements/1.1/":contributor, "http://purl.org/dc/elements/1.1/":date, "http://purl.org
/dc/elements/1.1/":type, "http://purl.org/dc/elements/1.1/":format,
"http://purl.org/dc/elements/1.1/":identifier, "http://purl.org/dc/elements/1.1/":source,
"http://purl.org/dc/elements/1.1/":language,
"http://purl.org/dc/elements/1.1/":relation, "http://purl.org/dc/elements/1.1/":coverage,
"http://purl.org/dc/elements/1.1/":rights}' is expected.'
```

```
</problem>
  </datastream>
</datastreamProblems>
</validation>
```

5.3 *Utility Methods*

5.3.1.1 **Upload** **URL Syntax**

/upload

HTTP Method

POST

HTTP Response

202 and a URI for the uploaded file

Parameters

Multipart file as request content

Examples

POST: /upload (with Multipart file)

III.VII Technote: Islandora

Namba L.: “BHL-Europe Islandora Reference Documentation”, 2011-04

Composed of excerpts from documentation taken directly from the Islandora website (<https://wiki.duraspace.org/display/ISLANDORA>).

ECP-2008-DILI-518001

BHL-Europe

Technical Note

BHL-Europe Islandora Reference Documentation

Deliverable number	<i>TN-Addendum-Islandora</i>
Dissemination level	<i>Public</i>
Delivery date	<i>2011-04</i>
Status	<i>Final</i>
Author(s)	<i>Lee Namba</i>



eContentplus

This project is funded under the *eContentplus* programme⁴⁶,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

⁴⁶ OJ L 79, 24.3.2005, p. 1.

Table of contents

1	DOCUMENT HISTORY	158
1.1	CONTRIBUTORS	158
1.2	REVISION HISTORY	158
1.3	DISTRIBUTION	158
2	PURPOSE OF THIS DOCUMENT	159
3	INSTALLATION AND CONFIGURATION	159
3.1	REQUIREMENTS	159
3.1.1	<i>Pre-installation software checklist:</i>	159
3.2	INSTALLATION	160
3.2.1	<i>Drupal Servlet Filter</i>	160
3.2.2	<i>The Islandora Module</i>	161
4	USING ISLANDORA	162
4.1	ISLANDORA COLLECTION OBJECTS	162
4.2	COLLECTION_POLICY	163
4.3	COLLECTION_VIEW	165
4.4	CHILD_SECURITY	173
4.5	QUERY	174
4.6	ISLANDORA CONTENT MODELS	174
4.7	COLLECTION & OBJECT ADMINISTRATION	175
5	APPENDIX	176
5.1	DRUPAL	176
5.2	FEDORA	177

1 Document History

1.1 Contributors

This document is a reference document and is composed of excerpts from documentation taken directly from the Islandora website (<https://wiki.duraspace.org/display/ISLANDORA>).

Person	Partner
Lee Namba	ATOS

1.2 Revision History

Revision Date	Author	Version	Change Reference & Summary
2011-04	Lee Namba	0.1	1. Draft
2011-05-03	Chris Sleep	Final	Dissemination changed to public for D3.7

1.3 Distribution

This document has been distributed to:

Group	Date of issue	Version

2 Purpose of this document

Islandora is an open-source framework developed by the University of Prince Edward Island's Robertson Library.

Islandora uniquely combines and harnesses the power of the Drupal content management system and the Fedora Digital Repository software to create a robust digital asset management system that can be fitted to meet the short and long-term collaborative requirements of digital data stewardship.

This document is a reference document and is composed of excerpts from documentation taken directly from the Islandora website (<https://wiki.duraspace.org/display/ISLANDORA>). We have edited the relevant text as it applies to the BHL-Europe system.

It aims to aggregate the essential Islandora documentation for developers and maintainers of the BHL-Europe system.

3 Installation and Configuration

3.1 Requirements

To successfully install Islandora, a site administrator should ideally have experience with configuring and trouble-shooting issues on a UNIX-based web server and with using UNIX command-line functions.

In addition, a site administrator should have or obtain a basic understanding of the following:

- Drupal (www.drupal.org)
- Fedora Repository (<http://www.fedora-commons.org>)
- Foxml 1.1 (<http://www.fedora-commons.org/documentation/3.0b2/userdocs/digitalobjects/introFOXML.html>)

3.1.1 Pre-installation software checklist:

The Islandora framework relies upon a number of other open-source applications. Before beginning the installation of any Islandora modules, ensure:

1. You have Drupal installed and properly configured with:
 - Clean URLs enabled in Drupal
 - The Drupal file system set to public
2. You have Fedora installed and properly configured:
 - Ensure you can use the admin tools in Fedora to ingest and purge.
 - A requirement for collection objects: To make the module more flexible and useful we have also made some specific decisions regarding our Fedora objects. For the module to be able to browse collections, your collection objects must have a hasModel entry in the RELS-EXT datastream that points to islandora:collectionCModel. This lets the module know that the object represents a collection and it will then query for objects that are members of this collection.
3. Other requirements beyond what is needed by Fedora and Drupal include:

- PHP5-curl
- PHP5-soap
- PHP5-xsl

It is advisable to review the help documentation if you are unfamiliar with these applications.

At the end of this installation, you will be ready to populate your site with digital assets and be capable of accepting Solution Packs.

3.2 *Installation*

3.2.1 **Drupal Servlet Filter**

The Drupal Servlet Filter allows the Fedora Repository to use Drupal's database for authentication and retrieving user roles.

3.2.1.1 **Installation Steps**

1. Download the latest version of the Drupal Servlet Filter from the Islandora github distribution site (<https://github.com/Islandora/Islandora-dist/>) and place it in \$FEDORA_HOME/tomcat/webapps/fedora/WEB-INF/lib

Ensure you choose the correct jar file for i) your version of Fedora, and ii) your authentication type (FeSL or legacy).

Note: If your Drupal and Fedora installations use different database types, Fedora will require the Drupal database driver's jar file in this directory as well. For instance, if Fedora uses postgres and Drupal uses MySQL, Fedora will require the MySQL jar file for the Drupal Servlet Filter in order to connect to the Drupal database.

2. Make the Fedora Repository aware of the new filter by following the instructions for Legacy Authentication:

3.2.1.1.1 **Legacy Authentication**

new xml elements must be added in order to configure Fedora's servlet filtering.

Edit the web.xml file located in \$FEDORA_HOME/tomcat/webapps/fedora/WEB-INF/ to include a reference to the Drupal Servlet Filter. Immediately after the <filter> element named XmlUserfileFilter, insert the following:

```
<filter>
  <filter-name>DrupalFilter</filter-name>
  <filter-class>ca.upei.roblib.fedora.servletfilter.FilterDrupal</filter-class>
</filter>
```

Then, immediately after the <filter-mapping> element named XmlUserfileFilter, insert the following:

```
<filter-mapping>
  <filter-name>DrupalFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```


3.2.1.1.2

3. Enable the Drupal Servlet Filter by creating the file filter-drupal.xml in \$FEDORA_HOME/server/config. Copy the following text as a template, then modify the attributes of the <connection> tag to match the server, port, database name, username and password of your site's Drupal database. Note: Fedora requires a separate <connection> entry for each connecting Drupal site.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!--File to hold drupal connection info for the FilterDrupal servlet filter. For multisite drupal installs you can include multiple connection elements. We will query all the databases and assume any user in any drupal db with the same username and password are the same user. We will gather all roles for that user from all databases. This is a potential security risk if a user in one drupal db has the same username and password as another user in a separate drupaldb. We are also assuming all drupal dbs to be mysql. This file should be located in the same directory as the fedora.cfcg file-->
```

```
<FilterDrupal_Connection>
```

```
  <connection server="localhost" dbname="drupaldb" user="dbuser" password="password"
port="3306">
  <sql>
```

```
    <!--Different sql statement for each connection. This is for drupal multisites that are setup
using one database with table prefixes. We don't do this but some people might.-->
```

```
      SELECT DISTINCT u.uid AS userid, u.name AS Name, u.pass AS Pass, r.name AS Role
FROM (users u LEFT JOIN users_roles ON u.uid=users_roles.uid) LEFT JOIN role r ON
r.rid=users_roles.rid WHERE u.name=? AND u.pass=?;
```

```
    </sql>
  </connection>
```

```
</FilterDrupal_Connection>
```

4. Stop and restart Fedora to enable the Drupal Servlet Filter.

5. Test the Drupal Servlet Filter by accessing your Fedora Admin client using your Drupal login credentials.

3.2.2 The Islandora Module

The Islandora module is a Drupal module written to allow the Drupal content management system to act as a front end to a Fedora Digital Repository. The module enables viewing and management of Fedora objects. This includes ingest, purge, add data stream, searching and browsing by collection. This version of the module does not store any data regarding any of the Fedora Objects in the Drupal database. The only data stored in Drupal is the configuration data telling Drupal how to connect to Fedora.

Future versions of the module may store metadata and/or links to datastreams regarding the Fedora object in the Drupal database. This would enable a Fedora-linked node to be used in the standard Drupal ways, but would also lead to the duplication of data and the problem of

keeping Drupal and Fedora in sync. The Islandora team is interested in ideas on how to make this work most efficiently.

To install the Islandora Module:

1. Download the latest version of the module from <http://www.github.com/islandora> and place the uncompressed contents of the module in your sites/all/modules or the sites/default/modules directory. For multi-site Drupal environments, refer to the [Drupal.org](http://drupal.org) instructions.
2. Enable the module by logging in to Drupal and navigating to Administration > Modules. Locate the module entitled Fedora Repository from the list of modules and enable the Digital Repository component of the module. Note: If there are missing dependent modules, ensure you have installed and enabled these to properly utilize Islandora.

You have now enabled the Islandora module. Navigate to your newly created Digital Repository menu item to view the objects from your Fedora Repository through your web site.

If no objects are found in your Digital Repository, you can quickly populate Fedora with some demo objects. To do this, go to Administer > Site Configuration > Fedora Collection List. Check the default information that populates the collection list form fields and your connection to the Fedora database. Then, select the Solution Packs tabs at the top of your page.

If you encounter problems with your Islandora configuration, check the following:

1. Your Fedora connection information is correct: The Fedora RISearch URL will, by default, specify localhost for your Fedora server name. If you are not using localhost, ensure you have entered your Fedora server's IP or domain name.
2. You have the appropriate user permissions to determine who can do what to Fedora objects from within Drupal. (need a bit more info on this)
3. The Fedora Default Display Object PID and Fedora Datastream ID are the defaults used by Drupal when it can't find a PID/datastream. Ensure these point to an object/datastream that is known to exist in your Fedora repository. Usually this will be an image indicating object not found or image not available. PID namespaces allowed in this Drupal install is a space-separated enumeration. Only Fedora objects identified by the members of this enumeration will be visible to users of this site. Similar to the retain PID namespaces in the older versions of Fedora config file.

4 Using Islandora

4.1 Islandora Collection Objects

An Islandora Collection Object is a Fedora object with several required datastreams. A familiarity of Fedora's object model is therefore essential to properly understanding the concepts underlying the creation and manipulation of collection objects in Islandora. The following documents offer a grounding in Fedora's Digital Object Model and Content Model Architecture:

- Fedora Digital Object Model:
<https://wiki.duraspace.org/display/FCR30/Fedora+Digital+Object+Model>
- Tutorial 2: Creating Fedora Objects Using the Fedora Content Model Architecture:
<http://www.fedora-commons.org/documentation/3.0b1/userdocs/tutorials/tutorial2.pdf>

In Islandora, collection objects must have a hasModel relationship to the islandora:collectionCModel and they must have a COLLECTION_POLICY datastream. This relationship tells Islandora that this Fedora object is a collection object. Islandora can then query the resource index for objects that have a relationship of isMemberOfCollection to this collection object.

The isMemberOfCollection is the default relationship used by Islandora, you can use others by specifying the relationship element in the collection policy xml. You would then have to store a QUERY datastream in the collection object.

The Collection Object defines four datastreams:

COLLECTION_POLICY
COLLECTION_VIEW
CHILD_SECURITY
QUERY

4.2 COLLECTION_POLICY

A Collection policy is an XML data-stream in a Fedora object with a DSID of COLLECTION_POLICY. The collection policy defines what content models may be ingested and related to this collection object. An example of a COLLECTION_POLICY may look something like this:

```
<collection_policy xmlns="http://www.islandora.ca"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
name="" xsi:schemaLocation="http://www.islandora.ca http://
syn.lib.umanitoba.ca/collection_policy.xsd">
<content_models>
<content_model dsid="ISLANDORACM" name="REFWORKS"
namespace="ir:ref" pid="islandora:refworksCModel"></
content_model>
<content_model dsid="STANDARD_PDF"
name="STANDARD_PDF" namespace="vre:ref"
pid="vre:contentmodel"></content_model> <content_model
dsid="ISLANDORACM" name="Collection"
namespace="islandora:collection"
pid="islandora:collectionCModel"></content_model>
</content_models>
<search_terms>
<term field="dc.title">dc.title</term>
<term field="dc.creator">dc.creator</term>
<term default="true" field="dc.description">dc.description</
```

```

term>
<term field="dc.date">dc.date</term>
<term field="dc.identifier">dc.identifier</term>
<term field="dc.language">dc.language</term>
<term field="dc.publisher">dc.publisher</term>
<term field="dc.rights">dc.rights</term>
<term field="dc.subject">dc.subject</term>
<term field="dc.relation">dc.relation</term>
<term field="dcterms.temporal">dcterms.temporal</term>
<term field="dcterms.spatial">dcterms.spatial</term>
<term field="fgs.DS.first.text">Full Text</term>
</search_terms>

```

40

```
</relationship></relationship>
```

```
</collection_policy>
```

The Collection Policy above would allow either of twotypes of objects to be ingested - STANDARD_PDF and REWORKS. How these are ingested and viewed is determined in the actual CONTENT_MODEL data stream.

Required for Ingest: The Islandora module requires a collection object to have a COLLECTION_POLICY datastream if additional objects are to be ingested as members of that collection object.

Adding a Collection Policy to a Collection Object

As described earlier, a Collection Policy is a data-stream (COLLECTION_POLICY) in a Collection or Parent-type Object that declares what other objects this object is allowed to be related to on ingest. It can optionally specify the relationship to use. If there is no relationship element, the default relationship will be isMemberOfCollection. Below is a snippet of the COLLECTION_POLICY stream that was added to the demo:SmileyStuff object:

```

<contentmodels>
<contentmodel name="STANDARD_JPEG">
<pid_namespace>demo:Smiley</pid_namespace>
<pid>demo:DualResImage</pid>
<dsid>ISLANDORACM</dsid>
</contentmodel>
</contentmodels>

```

41

```
<relationship>fedora:isMemberOf</relationship><!-- the demo
```

Smiley Stuff QUERY stream queries for isMemberOf

so we will use that relationship on ingest-->

Creating a COLLECTION_POLICY stream is similar to creating an ISLANDORACM stream in that you will have to create the XML by hand. It is probably best to start with an

example collection policy, edit the example and save it as a different name. You can use the Fedora Admin client to add the COLLECTION_POLICY stream or use Islandora itself by browsing to the Collection object, expanding the detailed list of content and adding the data-stream. The main requirement is that the data-stream have a dsid of COLLECTION_POLICY.

You can review the COLLECTION_POLICY datastreams in the collection objects that ship with Islandora.

Some additional examples that are available online:

Fraction COLLECTION_POLICY
https://wiki.duraspace.org/download/attachments/11502608/fractions_COLLECTION_POLICY.xml

Standard PDF COLLECTION_POLICY
<https://wiki.duraspace.org/download/attachments/11502608/PDF-COLLECTION+POLICY.xml>

4.3 COLLECTION_VIEW

The collection object's optional COLLECTION_VIEW data stream holds an XSLT to define how objects in that collection are displayed. Here is an example of a COLLECTION_VIEW:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" xmlns:s="http://www.w3.org/2001/sw/DataAccess/rf1/
result" version="1.0">
<!-- Red and White XSLT -->
<xsl:variable name="BASEURL">
<xsl:value-of select="$baseUri"/>
</xsl:variable>
<xsl:variable name="PATH">
<xsl:value-of select="$path"/>
</xsl:variable>
<xsl:variable name="thisPid" select="$collectionPid"/>
<xsl:variable name="thisTitle" select="$collectionTitle"/>
<xsl:variable name="size" select="20"/>
<xsl:variable name="page" select="$hitPage"/>
<xsl:variable name="start" select="((number($page) - 1) *
number($size)) + 1"/>
<xsl:variable name="end" select="($start - 1) + number($size)/>
<xsl:variable name="cellsPerRow" select="4"/>
<xsl:variable name="count" select="count(s:sparql/s:results/
s:result)/>
<xsl:template match="/">
```

```

<xsl:if test="$count>0">
<table cellpadding="3" cellspacing="3" width="90%">
<tr><td colspan="{ $cellsPerRow}">
<div STYLE="text-align: center;">
<xsl:choose>
<xsl:when test="$end >= $count and $start = 1">
<xsl:value-of select="$start"/>-<xsl:value-of select="$count"/>
of <xsl:value-of select="$count"/>&#160;<br />
</xsl:when>
<xsl:when test="$end >= $count">
<xsl:value-of select="$start"/>-<xsl:value-of select="$count"/>
of <xsl:value-of select="$count"/>&#160;<br />
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$thisPid"/>/-<xsl:valueof
select="$thisTitle"/><xsl:value-of select="$page - 1"/>
</xsl:attribute>
&lt;&lt;Prev
</a>
</xsl:when>
<xsl:when test="$start = 1">
<xsl:value-of select="$start"/>-<xsl:value-of select="$end"/>
of <xsl:value-of select="$count"/>&#160;<br />
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$thisPid"/>/-<xsl:valueof
select="$thisTitle"/><xsl:value-of select="$page + 1"/>
</xsl:attribute>
Next>>
</a>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$start"/>-<xsl:value-of select="$end"/>
of <xsl:value-of select="$count"/>&#160;<br />
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$thisPid"/>/-<xsl:valueof
select="$thisTitle"/><xsl:value-of select="$page - 1"/>

```

```

</xsl:attribute>
&lt;&lt;Prev
</a>&#160;
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$thisPid"/>/-<xsl:valueof
select="$thisTitle"/><xsl:value-of select="$page + 1"/>
</xsl:attribute>
Next>>
</a>
</xsl:otherwise>
</xsl:choose>
</div> <br clear="all" />
</td></tr>
<!--<xsl:for-each select="/sparql/results/result[position()>=$start
and position() &lt;=$end]">
<xsl:variable name='OBJECTURI' select="object/@uri"/>
<xsl:variable name='PID' select="substring-after
($OBJECTURI, '/')"/>
<tr>
<td>
<img>
<xsl:attribute name="src"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$PID"/>/TN
</xsl:attribute>
</img>
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:copy-of select="$PID"/>/-<xsl:value-of
select="title"/>
</xsl:attribute>
<xsl:value-of select="title"/>
</a>
</td>
</tr>
</xsl:for-each>-
-->
<xsl:apply-templates select="s:sparql/s:results"/>

```

```

</table><br clear="all" />
<div STYLE="text-align: center;">
<xsl:choose>
<xsl:when test="$end >= $count and $start = 1">
<xsl:value-of select="$start"/>-<xsl:value-of select="$count"/>
of <xsl:value-of select="$count"/>&#160;<br />
</xsl:when>
<xsl:when test="$end >= $count">
<xsl:value-of select="$start"/>-<xsl:value-of select="$count"/>
of <xsl:value-of select="$count"/>&#160;<br />
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$thisPid"/>/-<xsl:valueof
select="$thisTitle"/><xsl:value-of select="$page - 1"/>
</xsl:attribute>
<img alt="Previous page icon" data-bbox="113 448 155 462"/>
</a>
</xsl:when>
<xsl:when test="$start = 1">
<xsl:value-of select="$start"/>-<xsl:value-of select="$end"/>
of <xsl:value-of select="$count"/>&#160;<br />
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$thisPid"/>/-<xsl:valueof
select="$thisTitle"/><xsl:value-of select="$page + 1"/>
</xsl:attribute>
<img alt="Next page icon" data-bbox="113 678 155 692"/>
</a>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$start"/>-<xsl:value-of select="$end"/>
of <xsl:value-of select="$count"/>&#160;<br />
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$thisPid"/>/-<xsl:valueof
select="$thisTitle"/><xsl:value-of select="$page - 1"/>
</xsl:attribute>
<img alt="Previous page icon" data-bbox="113 903 155 917"/>

```



```

</a>&#160;
<a>
<xsl:attribute name="href"><xsl:value-of select="$BASEURL"/
>/fedora/repository/<xsl:value-of select="$thisPid"/>/-<xsl:valueof
select="$thisTitle"/><xsl:value-of select="$page + 1"/>
</xsl:attribute>
Next>>
</a>
</xsl:otherwise>
</xsl:choose>
</div>
</xsl:if>
</xsl:template>
<xsl:template match="s:sparql/s:results">
<xsl:for-each select="s:result[position() mod $cellsPerRow = 1
and position()>=$start and position() &lt;=$end]">
<tr>
<xsl:apply-templates select=". | following-sibling::s:result
[position() &lt; $cellsPerRow]" />
</tr>
</xsl:for-each>
</xsl:template>
<xsl:template match="s:result">
<xsl:variable name='OBJECTURI' select="s:object/@uri"/>
<xsl:variable name='CONTENTURI' select="s:content/@uri"/>
<xsl:variable name='CONTENTMODEL' select="substring-after
($CONTENTURI, '/')"/>
<xsl:variable name='PID' select="substring-after
($OBJECTURI, '/')"/>
<xsl:variable name="newTitle" >
<xsl:call-template name="replace-string">
<xsl:with-param name="text" select="s:title"/>
<xsl:with-param name="from" select="'_'"/>
<xsl:with-param name="to" select="' '"/>
</xsl:call-template>
</xsl:variable>
<xsl:variable name="linkUrl">
<xsl:choose>

```

```

<xsl:when
test="($CONTENTMODEL='islandora:collection')">
<xsl:value-of select="$BASEURL"/>/fedora/repository/
<xsl:copy-of select="$PID"/>/-<xsl:value-of select="s:title"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$BASEURL"/>/fedora/repository/
<xsl:copy-of select="$PID"/>/OBJ/<xsl:value-of select="s:title"/>
</xsl:otherwise>
</xsl:choose>
<xsl:value-of select="s:content"/>
</xsl:variable>
<td valign="top" width="25%">
<a>
<xsl:attribute name="href"><xsl:value-of select="$linkUrl"/>
</xsl:attribute>
<img>
<xsl:attribute name="src"><xsl:value-of select="$BASEURL"/>/
fedora/repository/<xsl:value-of select="$PID"/>/TN
</xsl:attribute>
<xsl:attribute name="alt"><xsl:value-of select="$newTitle"/>
</xsl:attribute>
</img> </a> <br clear="all" />
<a>
<xsl:attribute name="href"><xsl:value-of select="$linkUrl"/>
</xsl:attribute>
<xsl:value-of select="$newTitle"/>
</a>
<xsl:if test="($CONTENTMODEL!
='islandora:collectionCModel')">
<br />--<a>
<xsl:attribute name="href">
<xsl:value-of select="$BASEURL"/>/fedora/repository/
<xsl:copy-of select="$PID"/>/-<xsl:value-of select="s:title"/>
</xsl:attribute>
DETAILS
</a>--
</xsl:if>

```

```

</td>
<xsl:if test="(position() = last()) and (position() &lt;
$cellsPerRow)">
<xsl:call-template name="FillerCells">
<xsl:with-param name="cellCount" select="$cellsPerRow -
position()"/>
</xsl:call-template>
</xsl:if>
</xsl:template>
<xsl:template name="FillerCells">
<xsl:param name="cellCount"/>
<td>&#160;</td>
<xsl:if test="$cellCount > 1">
<xsl:call-template name="FillerCells">
<xsl:with-param name="cellCount" select="$cellCount - 1"/>
</xsl:call-template>
</xsl:if>
</xsl:template>
<xsl:template name="replace-string">
<xsl:param name="text"/>
<xsl:param name="from"/>
<xsl:param name="to"/>
<xsl:choose>
<xsl:when test="contains($text, $from)">
<xsl:variable name="before" select="substring-before($text,
$from)"/>
<xsl:variable name="after" select="substring-after($text,
$from)"/>
<xsl:variable name="prefix" select="concat($before, $to)"/>
<xsl:value-of select="$before"/>
<xsl:value-of select="$to"/>
<xsl:call-template name="replace-string">
<xsl:with-param name="text" select="$after"/>
<xsl:with-param name="from" select="$from"/>
<xsl:with-param name="to" select="$to"/>
</xsl:call-template>
</xsl:when>
<xsl:otherwise>

```

```
<xsl:value-of select="$text"/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

The Drupal Fedora module asks for the query to return SPARQL XML. So, a query like:

```
select $object $title $content from <#ri>
where $object <fedora-model:label> $title
and $object <fedora-model:hasModel> $content
and $object <fedora-rels-ext:isMemberOfCollection>
<info:fedora/demo:pid>
and $object <fedora-model:state> <info:fedora/fedora-system:def/
model#Active> order by $title
Would return results like:
```

```
<sparql xmlns="http://www.w3.org/2001/sw/DataAccess/rf1/
result">
<head>
<variable name="object"/>
<variable name="title"/>
<variable name="content"/>
</head>
<results>
<result>
<object uri="info:fedora/vre:ref-coll-188"/>
<title>A Test Collection</title>
<content uri="info:fedora/fedora-system:FedoraObject-3.0"/>
</result>
<result>
<object uri="info:fedora/vre:ref-coll-188"/>
<title>A Test Collection</title>
<content uri="info:fedora/islandora:collection"/>
</result>
<result>
<object uri="info:fedora/test:mark-library-1009"/>
<title>Building a Library 2.0 Tapestry</title>
<content uri="info:fedora/fedora-system:FedoraObject-3.0"/>
</result>
```

```

<result>
<object uri="info:fedora/test:mark-library-109"/>
<title>Building a Library 2.0 Tapestry</title>
<content uri="info:fedora/fedora-system:FedoraObject-3.0"/>
</result>
<result>
<object uri="info:fedora/test:mark-library-141"/>
<title>Coping With Change: Sys/Admin</title>
<content uri="info:fedora/fedora-system:FedoraObject-3.0"/>
</result>
</sparql>

```

You would use the XSLT as described above to transform the SPARQL XML to HTML. XSLT has to be matched to the Query. If you define a QUERY datastream your XSLT must be written to transform the results of that query.

4.4 *CHILD_SECURITY*

All objects in the collection will inherit the policies detailed in the collection object's CHILD_SECURITY datastream. This gives us security at the collection level.

All objects in that collection will have the same POLICY stream. If there is no CHILD_SECURITY stream at the collection level there will be no POLICY stream at the object level and, as such, Drupal permissions and global XACML policies will define the users with permission to modify this object.

Adding a Child Security Policy to a Collection Object

To enable Islandora to use XACML policies you will need to add a data-stream with a dsid of CHILD_SECURITY to any object that will act as a collection type object.

Creating and adding this stream is similar to the COLLECTION_POLICY and ISLANDORACM streams above. They will have to be created/modified by hand in a XML editor. Currently, the only way to add these streams is via Islandora's add stream form or the Fedora Admin client. As an example, to add security for the demo:SmileyStuff collection you would add an XACML stream with a dsid of CHILD_SECURITY to the demo:SmileyStuff object. This stream will then be added to all objects ingested into this collection as its POLICY stream. Caution: Be careful with POLICY streams as all access to an object can be lost if an object has an invalid POLICY stream.

We are also parsing the CHILD_SECURITY stream of the collection object to determine what users/roles can ingest at this level. This means that for now the XACML policies will have to be parse-able by our simple parser. Eventually, Islandora may include a callout determined by the Collection Policy to decide who can ingest in this collection.

Islandora XACML policies start out by denying access and then providing exceptions for users with certain roles or user ids. The file is parsed, looking for these roles/ids. If the user has any, they are allowed to ingest in the collection. An example XACML policy can be found on the DuraSpace Wiki1. This policy will allow all users to view but only the administrator role and fedoraAdmin user can modify. This could be the starting point of other policies and by adding users and roles determine who can modify the objects. A second

example2 provides an another starting point where all user access is blocked to all actions except to the users and roles listed.

Depending on your global XACML policies you may have to add a policy file to the \$FEDORA_HOME/data/ fedora-xacml-policies/repository-policies that will allow users who do not have the administrator role to ingest objects. This is only if you want non-administrator users to be able to manage objects. An example XACML policy3 that allows this available, but it opens API-M to all authenticated users.

If you have an XACML policy in every object that limits API-M this may be ok but you will probably want to modify this global policy to only allow certain roles to access API-M. By combining Drupal permissions and the Fedora XACML policies we hope to be able to keep the XACML relatively simple.

4.5 QUERY

A Collection object may have a QUERY datastream. If the object has a data-stream with a dsid of QUERY the Islandora module will attempt to use that query to get a list of objects related to that collection object. If there is no QUERY data-stream the module will try the generic one shown earlier in this document. Custom query/XSLT combinations should be written to expect SPARQL as the result. Here is a sample QUERY data-stream (would be uploaded with a text/plain mime-type):

```
select $object $title from <#ri>
where $object <dc:title> $title
and ($object <fedora-model:hasModel> <info:fedora/
islandora:mapCModel> or $object <fedora-model:hasModel>
<info:fedora/islandora:collectionCModel>)
and $object <fedora-rels-ext:isMemberOfCollection>
<info:fedora/imagined:collection>
and $object <fedora-model:state> <info:fedora/fedora-system:def/
model#Active>
order by $title
```

4.6 Islandora Content Models

Islandora uses Content Models to determine which mime-types can be ingested and how the object will be managed on ingest. This extends the Fedora Content Model Architecture (CMA).

Content models allow the definition of a custom data entry form to be displayed by the module for that object type. This allows differing data entry forms for differing object types.

The <display_in_fieldset> element determines how the object is displayed when a user accesses the object view.

The <ingest_rules> element defines how objects identified by specific dsid's are managed. For instance, a PDF content model may tell the module to create a thumbnail and ingest that thumbnail as an additional data-stream along with the actual PDF data-stream.

Islandora content models are stored as XML datastreams in a Fedora Content Model object with a datastream id (DSID) of ISLANDORACM. The collection policy data-stream, with a

DSID of COLLECTION_POLICY, references one or more content models defining what types of objects can be ingested in a particular collection. The Make Demo Smiley Stuff Islandora Aware4 page has some examples of Collection Policies and Content Models. Additional sample content models are linked below in the “Creating an Islandora Content Model” section.

In the Islandora content models we provide hooks that can be called at appropriate times, such as add datastream, edit metadata, ingest etc. The functions that are called by these hooks could then read more XML from the content model, for instance defining a data entry form. By using Islandora Content models you can make Islandora use the code that you provide. You can drop your php code into the modules directory (for example, under the plugins directory within the module) and then using the XML above you would be able to call your custom code or a combination of existing and custom code. Of course, you would require write access to the directory in order to copy your code.

4.7 Collection & Object Administration

If you have the appropriate Drupal permissions you will be able to ingest, purge and add datastreams. In Drupal your permissions are determined by the roles of the user you are logged in as. These permissions can be limited further by XACML policies. For instance, if you have a Drupal role that says you are allowed to add datastreams, you will be allowed to add datastreams to all objects except objects that have a XACML policy that denies it. View can also be blocked at the Fedora level using a XACML policy.

To manage objects in Fedora you browse to the object and, assuming the permissions allow, you can add/purge a data-stream, edit the metadata, or purge the object. Currently, you must edit raw XML to edit collection policies, content models and collection views. As an example, in order to change a collection view you would browse to the collection object, download the COLLECTION_VIEW stream, modify the XML, and then add the modified file back as a data-stream by clicking on the 'modify datastream' icon at the right of the datastream entry in the 'detailed list of content' section of an object's display page. There are some sample XML files shipped as part of the module. These files include Islandora Content Model (ISLANDORACM), Collection View (COLLECTION_VIEW), and Collection Policy (COLLECTION_POLICY) example files.

The simplest way to create an Islandora Content Model is to use the Islandora Content Modeler Module. The operation of this module is fully documented later in this guide.

Sample ISLANDORACM data-streams can be viewed in the objects that ship with the demo collections and others are available online.

Chemical Compound Content Model:

<https://wiki.duraspace.org/download/attachments/11502608/compoundcm.xml>

Specimen Content Model:

<https://wiki.duraspace.org/download/attachments/11502608/specimencm.xml>

5 Appendix

5.1 *Drupal*

Drupal is an open source content management platform powering millions of websites and applications. It's built, used, and supported by an active and diverse community of people around the world.

Pre-installation software checklist:

Drupal requires the following to be set-up and running prior to beginning your installation:

- Apache web server
- MySQL database are recommended.
- PHP 4 (4.3.5 or greater) or PHP 5 (<http://www.php.net/>)

*Installation Steps: *These are the quick “get-up-and-running” installation steps for Drupal. A more comprehensive installation guide is available from <http://drupal.org/documentation/install>

1. Obtain the latest Drupal release from <http://drupal.org/> and extract the contents of the compressed file. Note: Islandora is currently only compatible with Drupal 6.x.

2. Move the contents of the drupal-x.x directory into a directory within your web server's document root or public HTML directory (ensure that the .htaccess file, a hidden file, is successfully moved into the destination directory as well).

```
mv drupal-x.x/* drupal-x.x/.htaccess /var/www/html
```

3. Make a copy of the default.settings.php file in the sites/default directory and name the copy settings.php.

```
cp sites/default/default.settings.php sites/default/settings.php
```

4. Give the web server “write privileges” to sites/default/settings.php and the sites/default/ directory:

```
chmod o+w sites/default/settings.php
```

```
chmod o+w sites/default
```

5. Create a database for Drupal. Make note of your username and password as you will need it when the Drupal install script runs.

```
mysqladmin \-u <mysqlusername> \-p create <databasename>
```

```
mysql \-u <mysqlusername> \-p
```

```
enter your password
```

```
grant all on <databasename>.* to <db_user_name>@<server> identified by '<password>';
```

```
flush privileges;
```

6. Run the install script by pointing your browser to the base URL of your website (e.g., <http://www.example.com>).

7. Work through the on-screen steps to complete the Drupal site installation.

8. When the install script succeeds, you will be directed to the "Welcome" page, and you will be logged in as the administrator.

9. Proceed with the initial configuration steps suggested on the "Welcome" page.

For a good introduction to Drupal and to learn how to harness it's power and potential to create a site that meets your needs, access Drupal's extensive online documentation at <http://drupal.org/documentation> . An additional source of information is also Drupal's active open-source community, which can be accessed at <http://drupal.org/community> .

5.2 Fedora

Fedora or Flexible Extensible Digital Object Repository Architecture is a modular digital asset management (DAM) architecture that supports a variety of digital content needs. It was originally developed by researchers at Cornell University as an architecture for storing, managing, and accessing digital content in the form of digital objects. Fedora defines a set of abstractions for expressing digital objects, asserting relationships among *digital objects*, and linking "behaviors" (i.e., services) to digital objects. (ref: <http://www.fedora-commons.org/about>)

Fedora is available under the terms of the [Apache License](#) and has a very active open-source community producing additional tools, applications and utilities. At the time of this writing, Fedora 3.4.2 was the version available for download.

Pre-installation software checklist:

Fedora requires the following to be set-up and running prior to beginning your installation:

- Java SE Development Kit (JDK) 6: Available from <http://java.sun.com/>
- A database: Installed for Drupal. Consult the Fedora installation guide for notes on running other databases.
- An application server: Fedora includes the Tomcat Application Server. Consult the Fedora installation guide for notes on running other application servers.

Installation Steps:

1. Download the Fedora Repository software from <https://wiki.duraspace.org/display/FCR30/Installation+and+Configuration+Guide>
2. Read through the online guide to ensure the pre-installation system pre-requisites are met.
3. Preparing your local environment variables by modifying the `.bash_profile` or `.profile` file in the home directory of the fedora user.

The following example assumes Java is installed in `/opt/java` and Fedora is installed in `/usr/local/fedora`:

```
PATH=/opt/java/bin:$PATH:$HOME/bin
```

```
export FEDORA_HOME=/usr/local/fedora
```

```
export CATALINA_HOME=/usr/local/fedora/tomcat
```

```
export JAVA_OPTS="-Xms1024m \-Xmx1024m \-XX:MaxPermSize=128m \-  
Djavax.net.ssl.trustStore=/usr/local/fedora/server/truststore \-  
Djavax.net.ssl.trustStorePassword=tomcat"
```

```
export JAVA_HOME=/opt/java
```

4. Before beginning your Fedora installation, create a database for Fedora to use (In the install.properties file example that follows the database is called fedora3. This is referenced as part of the value string for database.jdbcURL).

5. To start the installer, navigate to the directory where the install file was downloaded and run the following a command:

```
java \-jar fcrepo-installer-3.4.2.jar
```

6. Select the CUSTOM INSTALL.

Note: It is important to select the Custom Install as it will enable the resource index by default, which is the backbone of Islandora's collection views and other functionality.

7. The Fedora installer script will ask you a series of questions. Once the script had collected your answers and configured Fedora on your system, the values are written to the install.properties file located in \$FEDORA_HOME/install.

An output of a sample install.properties file is included here to guide you through the installation.

An example of an install.properties [file](#):

```
\#Install Options  
\#Mon Dec 13 11:52:24 PST 2010  
keystore.file=included  
ri.enabled=true  
messaging.enabled=true  
apia.auth.required=false  
database.jdbcDriverClass=com.mysql.jdbc.Driver  
tomcat.ssl.port=8443  
ssl.available=true  
database.jdbcURL=jdbc:mysql://localhost/fedora3?useUnicode=true&characterEncoding=UTF-  
8&autoReconnect=true  
messaging.uri=vm:(broker:(tcp://localhost:61616))  
database.password=password  
database.mysql.driver=included  
database.username=fedoraUser  
fesi.authz.enabled=false  
tomcat.shutdown.port=8005  
deploy.local.services=true
```

```
xacml.enabled=true
database.mysql.jdbcDriverClass=com.mysql.jdbc.Driver
tomcat.http.port=8080
fedora.serverHost=localhost
database=mysql
database.driver=included
fedora.serverContext=fedora
llstore.type=akubra-fs
tomcat.home=/usr/local/fedora/tomcat
fesl.authn.enabled=false
database.mysql.jdbcURL=jdbc:mysql://localhost:8889fedora34useUnicode=true&characterEncoding=UTF-8&autoReconnect=true
fedora.home=/usr/local/fedora
install.type=custom
servlet.engine=included
apim.ssl.required=false
fedora.admin.pass=fedoraAdmin
apia.ssl.required=false
```

8. Once the installation script has completed and Fedora is installed, you should start your Fedora instance by running:

```
$FEDORA_HOME/tomcat/bin/startup.sh
```

9. To verify that Fedora has successfully started:

- a. `$FEDORA_HOME/tomcat/logs/catalina.out` should contain no errors.
- b. View your Fedora instance through a web browser: <http://localhost:8080/fedora/>

10. Access the Fedora Web Administrator: <http://localhost:8080/fedora/admin> and ensure you can ingest and purge objects.

11. For information on using Fedora, make use of the tutorials found at: <https://wiki.duraspace.org/display/FR22DOC/Fedora+Tutorials>