Architectural notes for the HOPE high-level design

Introduction

Purpose and scope

The high-level design (HLD) of the HOPE architecture identifies the basic concepts, principles, functions, data flows and open standards of the HOPE system.
The purpose of the high-level design is:

- to outline a technical approach to the HOPE infrastructure and system
- to provide sufficient guidance and direction for the project during the implementation of

the HOPE system.

The high-level design consists of the following documents:

1. HOPE Glossary
2. HOPE Vision
3. HOPE high-level design diagrams
4. Architectural notes for the HOPE high-level design

For a more general description of the vision and guiding principles behind the HOPE project and system, you can read the HOPE Vision document.
This document, entitled Architectural notes, is the compendium to the HOPE HLD Diagrams. It explains decisions taken for the high-level design of the HOPE System and the workflow of the diagrams. It identifies and analyses the architecturally significant requirements, constraints, decisions and objectives. It has a strong technical and design focus, but on a high level.
We have tried to find the right balance between HLD and describing the technical consequences of the choices made. However, we must stress that this document is neither a Functional Requirements nor a Technical Specifications document. It is not providing any comprehensive or detailed listing of the data requirements, conversion specifications, data flows, interfacing requirements, database requirements, etc. This document will serve as a starting point and a baseline for all the detailed specifications and deliverables planned later in the project (WP1, WP2, WP3, WP4 and WP5).

Incremental approach and Roadmap for implementation

Our approach to a high-level design conforms to the OpenUp practices, which we intend to follow during the design, development and implementation of the HOPE system. See for more information, the OpenUp wiki: Evolutionary Architecture.

The first version of the HOPE high-level design focuses on outlining the initial architectural decisions and forms the baseline on which HOPE work package 2 and the other work package tasks can start to build.
Work on the high-level design will continue:

- by gathering input from the best practices work packages (WP1 and WP2) and from the detailed implementation designs from work packages 3, 4 and 5
- by guiding the detailed implementation designs in order to ensure conformance to the high-level architectural decisions taken.

New iterations of the HLD-documents will be issued when necessary. Further refinement of the component parts of the architecture will be documented in separate documents, as part of the effort of work packages 4 (HOPE Aggregator) and 5 (HOPE Shared Object Repository).
All unresolved design and implementation issues are listed in the HOPE Design & Implementation Roadmap, which is an (internal) output of the high-level design task.

How to read this document

All readers should read the Glossary and Vision document before reading this document. This document should be read together with the HLD Diagrams in hand.
The current document is broken down into six main sections: The discovery-to-delivery process (d2d), the HOPE system and high-level workflow, the systems of content providers interfacing with HOPE, the HOPE Persistent Identifier (PID) Service, the HOPE Aggregator, and the HOPE shared object repository (SOR).

Section 1 identifies the major overall requirements for the HOPE system to realise the unambiguous and seamless navigation from a search result in any given discovery service to the digital resource (d2d logistic). [Vision, p.2]
Section 2 provides a high-level overview of the major functional components of the system and of the intended behaviour of the system to be captured in use cases.

Section 3 identifies the local systems of content providers (CP) which are required to interface with the HOPE system and it describes in some detail how the metadata and content of CPs can be integrated in the HOPE system.
Section 4 describes the high-level functionality of the HOPE Persistent Identifier Service and the main considerations why PIDs play such a critical role in the HOPE data-flow.
Section 5 describes the high-level functionality of the HOPE Aggregator, based on the existing D-NET system and highlights specific HOPE requirements to be implemented in D-NET.
Section 6 describes the high-level functionality of the HOPE Shared Object Repository and the main design considerations, components and API data-flows.

1. Discovery-to-delivery process

According to the Vision document the HOPE system is a web discovery-to-delivery service infrastructure, that will serve a variety of users, ranging from the scientific researcher to the general public, who want to find, access and use social history resources.
The journey between discovery and delivery of information resources on the Internet is accomplished with a variety of differing technologies and processes, many of which fall under the responsibility of different providers (discovery services, aggregators, repositories, etc.). In HOPE we need to take decisions concerning choices to improve d2d.

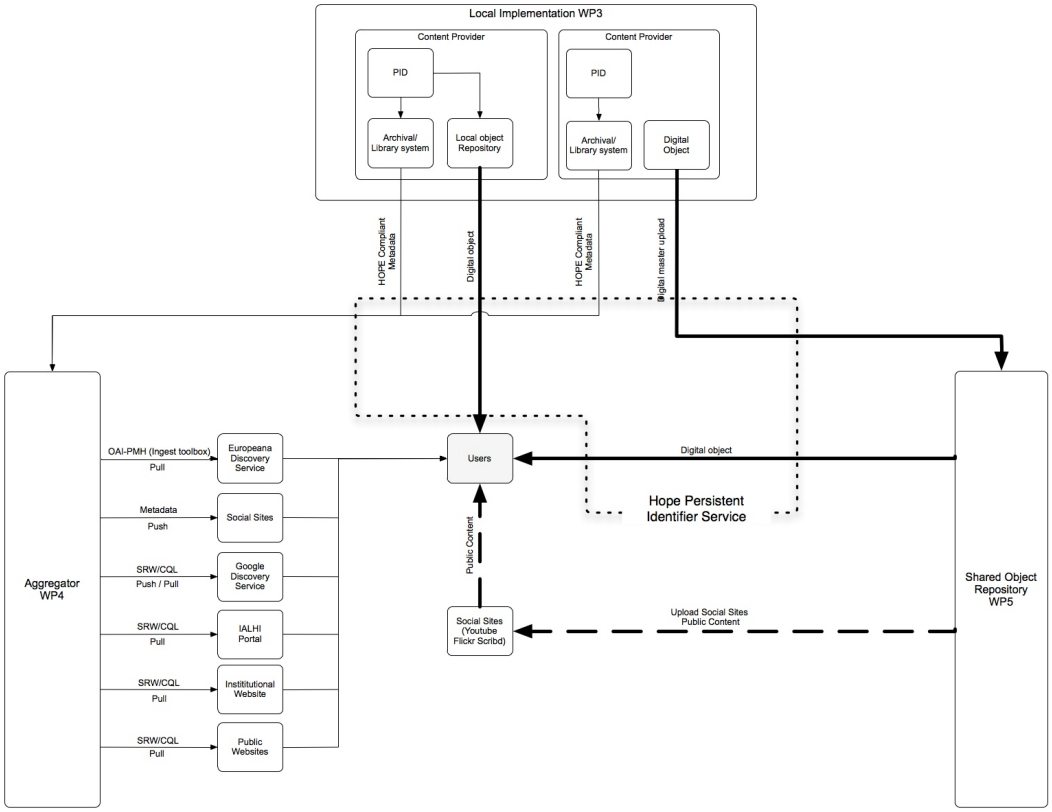
Important ways to ensure a seamless d2d process and to meet user needs and expectations are the use of:

1. *Open architecture principles for flexibility of infrastructure and interoperability of technical solutions;* the use of open standards ensures that there are as few dependencies as possible between various software components working together, and between different information systems that communicate with each other. The starting point for HOPE is to make maximal use of web-standards and standards for the exchange of data within the heritage sector and the research community (eg. SRU and OAI) - *furthermore* by limiting the choice of technological components to a number of proven open standards and open source solutions, it becomes possible to invest effectively in technological expertise and knowledge development within the HOPE community, minimising dependency on third parties with vendor lock-in implications.
2. *Best Practices for predictability and quality improvement of the d2d process;* there is inherently much variety in customer demand, hence the need for flexible service delivery to absorb that variety - but at the same time users also expect predictability in the way services are delivered, predictability in the way search services and delivery trajectories are offered - hence the need for harmonisation of practices and adoption of best practices.
3. *Internet protocol* (TCP/IP), in particular HTTP across all applications;
4. *Using URIs to uniquely identify information resources (digital objects and metadata) across the Internet* and beyond specific applications and information domains (e.g. loc-nr in WorldCat). In HOPE we adopt PIDs (i.e. persistent resolvable URIs) for all our digital objects and descriptive metadata units.
5. U*sing simple web API's to integrate with the HOPE components and other web-services.* HOPE learns from the best practices of the programmable web. Interaction with HOPE should be transparent for both the general web-user and application developers. Being part of the web is one of key design criteria for HOPE.
6. *Easy retrievability and use of public content,* without bothering users with online statements for the only purpose to protect CPs from any liability risk. The IPR guidelines (WP1) will work out these principles in more detail.
7. *Improving single sign-on authentication*: in HOPE we will not support single sign-on across d2d platforms. The starting point is that HOPE is an open, www-wide information space – it does not support closed systems. The assumption is that **any** web user may have found a link to an object file from the HOPE social history resource

from **any** discovery service. The HOPE compliant repository is discovery context-agnostic. The repository serves any web user requesting a digital object file, based on its Identification Authentication and Authorisation (IAA)-system. Community members of HOPE Partners can access the objects via API keys provided to the HOPE partners. The use of API keys ensures that if they want to display restricted objects on their local websites they can do this without additional login steps or cumbersome merging of login information.

2. HOPE system and high-level HOPE data-flows

The HOPE system consists of the local systems of Content Providers, the HOPE Aggregator, the HOPE PID service, the HOPE SOR and the discovery services.
Below a diagrammatic representation of the component parts of the HOPE system and of the data-flows can be found.



This section describes briefly the key components and identifies a number of key actors, the main use cases, and supporting requirements for the HOPE system as a whole. The basic flow of each of the use cases is outlined briefly and not described in any detail. This needs to be done in next iterations and the resulting use cases will be documented in other documents. However, it is important to note that in the HOPE data-flow the digital masters must be submitted to the SOR before the descriptive metadata can be submitted to the Aggregator.

The HOPE system consists of 3 core components:

1. The HOPE PID Service: which provides resolvable PIDs for the HOPE digital objects and metadata records and thereby ensures the d2d process;
2. The HOPE Aggregator: which integrates and harmonises all the metadata records of the HOPE CPs and disseminates the metadata to web discovery services;
3. The HOPE SOR: which keeps the digital masters of collections held by the HOPE CPs and delivers copies and derivates on demand;

These core components interact mainly with 4 types of actors in the environment surroundings:

1. Local systems of Content Providers: systems with which the CPs produce/manage their metadata and their digital collections;
2. Discovery Services: which provide search and browse facilities to discover the integrated content from Aggregators and other information resources on the web.
3. The web user: who makes use of the Discovery Services and of the delivery services of the SOR.
4. Social sites: which facilitate the sharing of digital content with/between web users and are generally focused on one type of content medium (text, video, pictures, music, etc.) or networking purpose (research, swimming, blogging, gaming, etc).

Main use cases:

1. CP supply-use-cases:
    a. Digital master upload by the CP to the SOR: The digital objects supplied to the SOR may come from any local network-accessible system used to store and access digital objects (eg. a digital assets management system, an FTP/HTTP-server or even a HOPE compliant repository). If the CP stores digital objects on offline carriers, he can choose to set-up a local network-accessible system for the supply of digital objects or to upload digital objects directly to the staging area of the SOR. The digital objects supplied should conform to the agreed HOPE standards, in particular they should each come with a PID.
    b. Supply/Harvesting of HOPE compliant metadata from the CP to the Aggregator: A system is required for assembling the exports from the Archival/Library system (the bibliographic records and archival finding aids) into metadata batches (data-sets), ready to be collected by the HOPE Aggregator. This system might be an OAI-PMH repository or an FTP-server for example. The metadata should be HOPE compliant, in particular they should contain the PID of the descriptive record and the PID of the corresponding digital object. This PID may be allocated by the local PID Service or fetched from the HOPE PID Service.
2. Aggregator export-use-cases:
    a. Metadata export from the Aggregator to Europeana: this export conforms to the Europeana Metadata Schema and to the export protocol supported by Europeana (OAI-PMH harvest and ingest into the Europeana Ingest Toolbox).
    b. Metadata export from the Aggregator to Google: this flow ensures that the HOPE metadata are supplied to the Google crawlers.
    c. Metadata export from the Aggregator to the IALHI Portal: this is the API interface of the Aggregator Search webservice.
    d. Metadata export from the Aggregator to the CP institutional websites: this is the API interface of the Aggregator Search webservice.
    e. Metadata export from the Aggregator to Public websites: this is the API interface of the Aggregator Search webservice.
3. User request-use-cases:
    a. User requests a digital object from the local HOPE compliant repository: a web-user who has found a relevant description of a digital object follows the digital object's PID link which activates a chain of HTTP-requests via the PID-resolver mechanism of the CP's local PID Service. The local HOPE compliant repository

reacts to the HTTP-request by providing the jump-off page with links to different versions of the digital object (a choice of derivatives) and/or with transactional information for accessing the resource (rights/payment transactions).

b. User requests a digital object from the SOR: a web-user who has found a relevant description of a digital object follows the digital object's PID link which activates a chain of HTTP-requests via the PID-resolver mechanism of the HOPE PID Service or the CP's local PID Service. The HOPE SOR reacts to the HTTP-requests by providing the jump-off page with links to different versions of the digital object (a choice of derivatives) and/or with transactional information for accessing the resource (rights/payment transactions).

c. User requests public HOPE content from the social site: a web-user who has found a relevant link to a digital object from the public HOPE content on social sites, follows the link which activates the social sites rendering services (eg. a video player), through which the digital object is displayed.

4. Upload of public content to social sites: this is the flow of well-defined sets of public content which the HOPE SOR uploads to a given social site (for which HOPE has opened an account). The returned embed-URLs are added to the jump-off page information and to the CPs. Additional descriptive metadata for the description is acquired from the Aggregator Search webservice and is pushed to the social sites together with the digital objects.

3. The systems of content providers interfacing with HOPE and data-supply flows

Section 2 permits us to identify the highest value and highest risk items so that we can concentrate on these first. During previous iterations of the High Level Design it became apparent that the design made some assumptions about the prerequisites for supply of data by CPs to the Aggregator (WP4) and to the Shared Object Repository (WP5). For WP3 and the first Best Practice Network Workshop to be held beginning of September 2010, it is very important to clarify up-front and as soon as possible the requirements for the Content Providers.

Our working assumption is that there are a set of minimal requirements which all CPs can fulfill, namely that the CPs:

1. have links between their metadata records and their digital objects,
2. are able to export their digital objects to some kind of file-system before uploading them to the SOR
3. are able to export their metadata in XML (encoded in UTF-8) and assemble the metadata in data-sets on some kind of file-system for the Aggregator.

In addition, our starting point is that CPs are joining the HOPE BPN with the intention to adhere to the HOPE best practices (see also Vision, Objectives) - namely in terms of:

1. providing HOPE compliant metadata,
2. applying persistent identification to their objects and metadata records
3. following agreed HOPE supply-procedures and work-flows
4. supporting agreed protocols to interface with the HOPE system.

CP data-supply flows

This sub-section explains the CP supply-use-cases in some detail, highlighting the requirements. The metadata/content requirements (eg. HOPE Metadata Schema, Content formats) need to be worked out in detail by WP2 and the use cases need to be worked out for each CP separately in the framework of WP3, but this section gives an illustration of the typical use case, with the various steps necessary to supply the descriptive metadata and the digital objects of a data-set in the HOPE system.

- Preparation by the content provider before submission of a data-set:
    - Add a persistent identifier to every descriptive metadata record. The PID must be coded in a (additional) metadata field of the metadata record in the Content Provider's own metadata management system (eg. archival or library information system). This identifier will be used by the Aggregator to identify, store and update the metadata record.
    - Add a persistent identifier for each digital (master) file (e.g. scan of a book page) that is part of the digital object (e.g. the book) corresponding to the relevant metadata record. One digital object (ex. a book or a letter) could consist of very many digital files. Each file should receive its own PID, but there is only one description for one object. So in the description of the object you want has only one PID, that of the structural metadata-file (e.g. a METS document) in which the different scans are structured for viewing. The persistent identifier of the container will be used by the SOR to ingest/store/ update the digital object and to disseminate derivatives. During ingestion, the SOR will update the resolvable links for the PIDs.
    - The content provider must be able to export the digital masters from his native store as files and submit the files paired with the persistent identifier.
    - Define a data-set: a set of descriptive metadata records that the CP wants to treat as a submission unit for the Aggregator.
    - The content provider must be able to export all descriptive metadata records as valid XML records and encoded in UTF-8. Preferably in a single file per data-set. Large files must be compressed (e.g. gzipped) to reduce network overhead.
    - The content provider should make the descriptive metadata available to the aggregator via OAI-PMH or FTP.
    - The content provider is responsible for pushing his digital objects to the Shared Object Repository first and then to allow the aggregator to pull the corresponding data-set with metadata records. The pairing is done on the metadata level. The CP is responsible for including the PID of the digital master or compound object in each metadata record provided to the Aggregator.
- Submission of digital masters to the Shared object repository
    - Initial submission
        - CP: Content Provider creates a XML processing instruction file that contains all the necessary information to make the submission API call to the SOR. This information includes amongst others: the persistent identifier, mime-type, access information, temporary location of the digital object for ingest (this can be on local disk, URL, or as part of the HTTP post), API-access key, content checksum of the digital object
        - CP: The Content Provider uses the SOR submission tool to process XML processing instruction and submits the digital objects to the SOR.
        - SOR: When the object is ingested the SOR ingest platform will update the resolve URL for the persistent identifier in the respective persistent identifier service to resolve to the SOR dissemination API.
    - Re-submission (updating)
        - Updating uses the same mechanism as the initial submission. Only the content checksum will be used to check if the same digital object is already stored. In that case only the technical metadata is updated. When the content hash is different the old digital master is disconnected from that PID and deleted providing no other PIDs are connected to it.
    - Deleting digital masters
        - For deleting digital masters again use the XML processing instruction. The API call then issues a delete request and changes the resolvable URL associated with the persistent identifier that is deleted to no longer point to the SOR.

- Submission of descriptive metadata to Aggregator
    - Initial submission
        - CP: Register as Provider with the Aggregator
        - CP: Register the data-set with the Aggregator
            - this includes OAI-PMH or FTP information so the Aggregator can harvest the descriptive metadata
        - CP: Create the metadata mapping following the BPN guidelines.
        - CP: Submit the mapping file to the Aggregator
        - Aggregator: The maintainers of the Aggregator software are responsible for transferring the mapping rules into the Aggregator.
        - CP: Flagging the data-set for first ingestion run in the Aggregator management interface
        - Aggregator: The Aggregator harvests the data-set and inserts the metadata records with the persistent identifiers.

      After the initial harvest, all records are available in the Aggregator for exposure to discovery services, in the metadata formats supported by the Aggregator
    - Re-submission (updating)
        - The Aggregator will periodically check for updates and re-ingest the data-set. New records are inserted, modified records are updated, and deleted records are flagged as deleted. For purposes of re-aggregation (eg. in Europeana) the PID of the deleted records is never deleted from the system, but just flagged as deleted. When at a later stage it is resubmitted, it will be flagged as active again.
        - During re-submission the data-set will not be locked, but the latest version is still available for re-aggregation/exposure.
    - Deleting metadata records
        - When either the OAI-PMH protocol is not properly implemented (i.e. without persistent deletions) or the metadata is delivered via FTP, a mechanism must be defined in the Aggregator to handle deleted records. For example, all records that were not modified after a re-submission will be flagged as deleted.
- Dissemination:
    - The Aggregator is able to disseminate the metadata records in a number of different metadata formats and different methods of access to the metadata - such as OAI-PMH, SRW, and plain export - both on record and data-set level.
    - All access to the object stored in the Shared Object Repository will go via the PID of the digital master that is supplied during ingestion.

HOPE compliant repositories

CPs can choose to make their digital content available through a local repository system, but in order to fit in the HOPE system, this local system needs to comply to a minimum set of requirements. The local system may be a a local instance of the SOR, in which case it will be HOPE compliant, but it could also be a newly acquired repository, or an enhanced version of the local systems. In such cases, the requirements for HOPE compliancy need to be made explicit. This will be worked out in more detail in the next iteration of the HLD (see Design & Implementation Roadmap).

4. HOPE Persistent Identifier Service

The HOPE BPN has chosen to require Persistent Identifiers for all metadata records and digital objects submitted to the HOPE system (see HOPE Vision Document). The usage of

these PIDs is very important to the maintainability of the HOPE system. Duplicates' detection and merging has been one of the core problems of large-scale aggregation and unification of heterogeneous sources. The consistent implementation of persistent identifiers by Content Providers will significantly simplify the work-flow and enhance the reliability of the HOPE system.

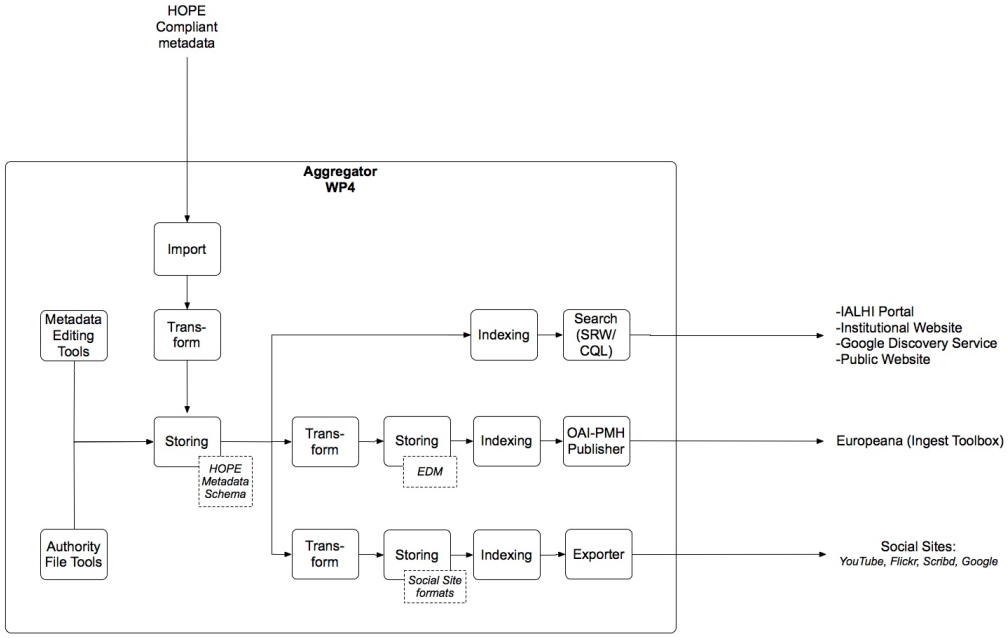
The PIDs submitted to HOPE must be registered at a known persistent identification web-service, such as HANDLE, PURL, DOI, ARK, etc.. WP2 will provide recommendations concerning which PID services conform to the HOPE requirements. The addition of persistent identifiers as resolvable URLs to digital objects will also give Content Providers much flexibility and control over the resolvability of the objects they refer to. For example, when the CP decides to migrate the digital objects to another repository they will only have to change the resolve URL that the PID refers to in the respective web-services. It will not be necessary anymore to issue new URLs for all these objects to all the users and services that are using or have bookmarked these objects.

The HOPE system will also provide its own Persistent Identifier Service, because not every Content Provider is able to host its own system. Since there are many PID services available on the web today, the HOPE BPN will choose one of these PID services as its preferred PID mechanism. The HOPE System will host a dedicated resolver for this preferred service. The CP that wants to make use of this HOPE service must first register an institutional PID at this preferred PID service, before they can start using the HOPE PID service. Next to hosting the PID service, HOPE will also provide tools to help CP add PIDs to their objects and metadata records (for example, a small stand-alone command-line tool that will simplify the addition of PIDs to objects/records in the local system with a JDBC compliant database). When the CP adds a dedicated PID field to their metadata, this tool will automatically populate these fields and register the PID at the HOPE Persistent Identifier Service.

So due to work-flow concerns the consistent usage of PIDs in the local CP system is deemed preferable. This solves many long-term integration problems for aggregation projects like HOPE. These issues are amongst others: duplicates' detection, dealing with orphaned items due to changed identifiers, problems with incremental updates, broken links throughout the system, etc. The PID based work-flow will also remove a lot of architectural bloat from the HOPE system and make the high-level work-flow much more transparent.

5. HOPE Aggregator

This section describes the D-NET Toolkit on which the *Aggregator* will be based and identifies and analyses the functionalities to be used in HOPE. Below a diagrammatic representation of the module can be found.



The D-NET Toolkit

The Aggregator module is realised by means of the D-NET Software Toolkit (for more info: http://www.d-net.research-infrastructures.eu).
The D-NET Toolkit offers a service-oriented framework where developers can build applications by combining a set of D-NET services.
Furthermore, the framework allows for the addition of new service typologies, in order to introduce new functionality, whenever this is required and without compromising the usability of other components. D-NET provides a *data management service kit*, whose services implement functionality for the gathering, manipulation and provision of XML records exported by a set of content providers. Such services have been realised and added in the years to meet the particular requirements arisen when facing new challenges in different application domains (e.g., DRIVER project, EFG project, OpenAIRE project). Most importantly, they are designed according to two engineering principles:

Modularity: services provide minimal functionality and exchange long lists of information objects through the ResultSet mechanism (by relying on a ResultSet Service instance or by implementing natively the interface functionality of the ResultSet Service) so that they can be composed with others to engage in complex data management workflows.
Customizability: services should support polymorphic functionalities, operating over XML records whose data model, i.e., XML format, matches a generic structural template. For example, the D-NET index service is designed to be customizable to index records of any XML format.

As mentioned above, the D-NET framework enables the combination of services into workflows, to obtain complex and personalised data processing operations. To this aim, the services are designed to exchange XML records through mechanisms offfered by D-NET ResultSet Services. The service manages *ResultSets*, i.e., “containers” for transferring list of files between a “provider” service and a “consumer” service. Technically, a ResultSet is an ordered lists of files identified by an *End Point Reference* (called EPR, the Web Service EPR standard describes the location of a resource on the Internet), which can be accessed by a consumer through paging mechanisms, while being fed by a provider. D-NET services can be designed to accept or return ResultSet EPRs as input parameters or results to invocations, in order to reduce response delays and limit the objects to be transferred at the consumer side to those effectively needed. For example, while the response to a full-text query may consists of thousands of rank-sorted results, the consumer often requires to access tens of them.

D-NET services for realising the HOPE Aggregator

The HOPE Aggregator has three main functional objectives:

1. collecting the data from content providers (harvesting, transformation and storage);
2. curating the records (editing, cleansing, enrichment);
3. disseminating the records to third-party systems (pull or push)

In the following sub-sections we describe the D-NET services to be used for the realisation of the three objectives and the implementation of the HOPE Aggregator. We shall see that, in order to satisfy HOPE requirements, in some cases D-NET services will have to be extended and new ones will have to be realised.

Collecting the data

Typically, the content providers will export their metadata in the form of XML records through OAI-PMH protocol APIs. The Aggregator has the task of administrating a set of “authorised” content providers in order to harvest their records, if necessary transform them into the HOPE metadata schema, and store them locally. To this aim, the following D-NET services will be combined into a data-set workflow:

Content Provider Manager Service. Content providers are “registered” to the Aggregation system, with a “profile” that describes their location and typology (currently “OAI-PMH compliant” and “remote FTP folders”). The service offers user interfaces for the registration (by content provider administrators) and the subsequent administration of the content providers and their data-sets (by HOPE aggregator administrators, who can fire harvesting, manage transformation mappings, etc).

Harvester Service. An Harvester Service can execute the six OAI-PMH protocol verbs, and communicate with a given data source registered to the system. In particular, the verb `ListRecords` fetches from the data source the metadata records of a given metadata format (e.g., oai_dc) and returns the EPR of a ResultSet that contains them.

File downloader Service. A File Downloader Service can import all XML files in a given local/ remote file system folder. In particular, a “download” call returns an EPR of a ResultSet that contains such files.

Transformator Service. A Transformator Service is capable of transforming metadata records of one data model into records of one output data model. This service is responsible for harmonisation of the metadata records. The logic of the transformation, called “mapping”, is expressed in terms of a rule language offering operations such as: (i) field removal, addition, concatenation and switch, (ii) regular expressions, (iii) invocation of an algorithm through a

Feature Extractor Service, or (iv) upload of full XSLT transformations. User interfaces support administrators at defining, updating and testing a set of mappings. A transformation request is thus composed by: input metadata format, EPR of input ResultSet, output metadata format, reference to the mapping to be applied. If the mapping is not available, the transformation is left pending until HOPE aggregator administrators will provide one. The result of a transformation is the EPR of a ResultSet that contains the generated metadata objects.

MDStore Service. HOPE XML metadata records collected (or obtained by transforming collected records) from content providers are aggregated into MDStore Services. An MDStore (factory) Service manages a set of MDStore units capable of storing metadata objects of a given metadata format. Consumers of the service can create and delete units, and add, remove, update, fetch, get statistics on metadata records from-to a given unit. A given MDStore unit is fed by passing the EPR of the ResultSet containing the incoming records, and when accessed returns an EPR to the ResultSet containing the output records.

Curating Authority Files and metadata

HOPE administrators (often a group of "experts in the field" selected across the content providers) may be willing to perform further semi-automatic cleansing activities. To this aim, the following D-NET Services will be used:

Authority File Service. The Authority File Service implements functionality for "curating" a set of *authority files*, intended as sets of authoritative metadata records. Note that authority files are typically kept, fed, administrated separately from the core data that depend on them, which has to be kept synchronised to the possible updates occurring to the authority files of reference. In particular, administrators can:

- create an authority file, by providing the relative metadata data model,
- create, delete, edit metadata records in it,
- use algorithms for the identification of candidate pairs of duplicate records,
- "merge" a pair of candidate records into one, and "split" metadata records, i.e., obtaining two records from one.

When finished, administrators can "commit" changes and generate a new version of the authority file. Each version is accompanied by a *report file*, which contains the list of merge or split operations committed in the file and that can be exploited by consumers (D-NET services or external applications, see Metadata Editor Service) to upgrade the data in data-sets making use of the authority file according to the latest updates. Consumers can feed an authority file with new metadata records (by sending an EPR of a ResultSet with new metadata records) or access an authority file, which is returned into a ResultSet through an EPR.

Note: *which sets of XML metadata records and which subparts of such records will be identified as authoritative in HOPE will be decided once the HOPE Metadata Schema will be defined.*
Extra: *authority files, as well as report files, may be accessed and exploited also by content providers to improve the quality of their data. Authority files will need to make use of PIDs in order to be accessible in an unequivocal fashion across the HOPE system and beyond.*

Metadata Editor Service. The service offers functionality for the manual or automated editing of the XML metadata records stored into the MDStore Service units. Manually, HOPE administrators can search, add, remove and edit the records in MDStores. Automatically, the report files from Authority Manager Services are processed so that changes can be propagated to the records involved.

Extra: An additional functionality will have to be implemented, to satisfy the following HOPE requirement: the HTTP address of the thumbnails generated by the HOPE Shared Object Repository will have to be added to the HOPE XML metadata records describing the relative objects in a second stage, through a remote request. The functionality will enable authorised applications to invoke an update operation (possibly bulk) over one given field of one given record (by identifier) with one given value (likely the functionality will be generalised to an arbitrary number of fields of the same record).

Disseminating the metadata

HOPE XML metadata records, which are continuously collected and curated as explained above, will be disseminated through different protocols and possibly different XML formats. To this aim, several service workflows will be constructed, one for each typology of data export. Whenever the data requires a transformation into another format, the Transformator Service will be involved: this is the case for the Europeana Metadata Schema (EDM) and for the records to be exported to third-party systems, which generally accept records matching specific import formats. In addition, the following D-NET services will be used:

Index Service. An Index (factory) Service manages a set of Index units capable of indexing metadata records of a given data model, i.e., XML format, and replying full-text CQL queries (*Contextual Query Language*, http://www.loc.gov/standards/sru/specs/cql.html) over such objects. Consumers can feed units with records, remove records or query the records. The Index Service replies to CQL queries by returning the EPR of a ResultSet that contains the result. Moreover, the service supports advanced full-text highlighted search and faceted browsing functionality. The Service is implemented on Solr (*Solr Apache Lucene Project,* lucene.apache.org/solr/).

Search Webservice. A Search webservice offers an SRW/CQL (*Search/Retrieval via URL*, http://www.loc.gov/standards/sru) interface accepting a CQL query Q and a metadata format, to run Q over the Index units matching that model. To this aim, queries are routed to the right Index Services; the responses, when more than one Index Service is involved, are then “fused” and pushed into a ResultSet, whose EPR is returned as result. Note that for performance reasons the Search webservice memorises a cache of the Index units available, kept up to date by subscribing to the creation and removal of Index units in the system.

OAI-PMH Publisher Service. An OAI-PMH Publisher Service offers OAI-PMH interfaces to third-party applications (i.e., harvesters) willing to access metadata objects in the MDStore units. To this aim, the service dynamically discovers and exposes through the `getFormats` verb the list of metadata formats currently available in MDStore units and offers a `getRecords` operation over all the MDStore units hosting such records.

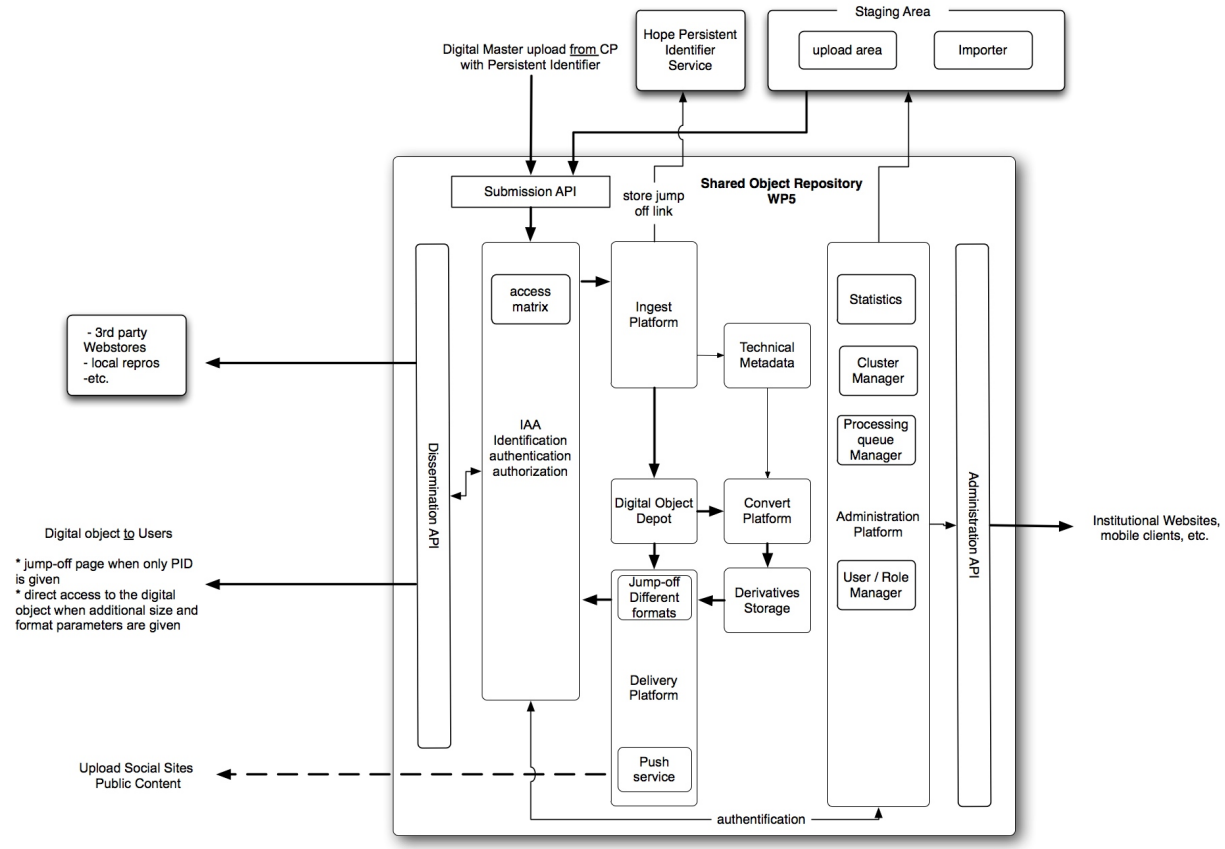
Export Service. **A new D-NET Service** will have to be realized, capable of exporting XML records from MDStore units to known web sites, such as YouTube, Flickr, Google and Scribd. For certain export services interaction with the SOR is required. Through the SOR dissemination API, the export service is able to determine which derivatives for a digital master is available and can determine the most appropriate format to be exported to the external websites. Preferably, the Export Service will not fetch the derivatives from the website, but construct the appropriate SOR url for external site to retrieve the derivative.

Metadata Synchronisation Service. Having a continuous feedback loop between the Aggregator and the content providers is one of the value propositions of the HOPE system. Especially, named entity recognition, the addition of harmonised geographical references, data harmonisation, and replacing literals with authoritative URLs are important to CPs. **A new D-NET service** will realise this feedback loop and make it possible for HOPE and the CPs to

become an integral part of the Linked Open Data cloud.

6. HOPE Shared Object Repository

This section describes the design considerations and proposed components for the Shared Object Repository. The SOR plays a critical role in the d2d process to make access to the digital masters and their derivatives more transparent to the user. In the future, the SOR can also play a critical role in the digital preservation of the digital masters.
Below a diagrammatic representation of the Shared Object Repository can be found.



Design considerations

- Descriptive metadata (of the digital objects) are for search and discovery. They are not functional or necessary for the operation of the repository. These metadata are produced and managed outside the digital object repository. The repository is descriptive metadata-agnostic.
- In order to promote the open web character of the SOR, all interaction with repository should go through Web-based APIs. Currently, three access types are identified: Submission, Dissemination and access to the Administration information.
- Converting digital master files to derivatives (i.e. different formats and sizes) is a function of the repository. The Aggregator is also capable of doing this function, but the Aggregator needs to fetch the digital master file from the repository first. It is better if the Aggregator gets the preview from the repository. Based on the availability of the PID in the descriptive

metadata, the Aggregator should be able to construct the dissemination API URL for thumbnail retrieval to the metadata in Aggregator storage.
- Streaming **previews** of films will not be supported by the repository. The previews will be put on YouTube and the link provided to the content provider and/or aggregator. The jump-off page if applicable will also contain links to the previews stored outside the SOR.
- Streaming of **digital objects** (master or derivative) will not be supported by the repository. Once a user decides he/she wants to see the whole film (after having seen the preview), a download copy will be made available to the user via the delivery platform.
- The repository will not natively contain a payment module. In order to recover cost for maintenance and organisational operation of the HOPE Repository interaction with third party web stores and local reproduction departments is envisioned. These parties would interact with the Dissemination API to acquire an digital object from the delivery platform after the payment has been completed. A scenario where a temporary link is created by the SOR to download the object via the Delivery Platform is also considered. The access to this temporary link will be managed through the dissemination API.
- The SOR is expected to grow rapidly in the coming years both in number of objects and number of users. Therefore scalability and the ability to replicate is of paramount importance. The content of the three storage components - Digital Object Depot, Technical Metadata Storage, and Derivative storage - should be easily replicable to different nodes in the cluster. In addition, the use of a Content Delivery Network provider should be investigated for the license-free objects to reduce the load and bandwidth requirements on the main SOR. Alternatively -- because of the single access design principle through the dissemination API -- the use of geographically distributed caching proxies is also considered to reduce the load on the main SOR. The later option would transparently interact with the IAA module.
- The SOR should be able to run as the master in a cluster - like it would in the HOPE System - but it should also be able to be installed on the local infrastructure of the content provider as a stand-alone system. Ideally, a remotely installed SOR should be able to be added to the cluster of the central HOPE Repository streamlining the synchronisation even further. This requirement is primarily geared towards enabling the CP to have their own digital repositories. This is a wish that is shared by many CPs in the HOPE network.

Shared Object Repository Components (SOR)

The following sections contain a short description of the functionality of each component in the SOR diagram.

APIs

The three public APIs are the only access mechanism to the SOR. Through the use of simple XML based APIs we hope to make the use and integration of SOR in existing workflows as easy as possible. Three access types have been identified: submission, dissemination and administration. Since the APIs are web-based it means they will take full advantage of asynchronous and concurrent processing of requests.

Submission API

The submission API is responsible for receiving a submission request for storing a digital master in the SOR. The XML processing instruction also contains an option to send a delete request for the digital master. See also Appendix: Sample XML Processing Instruction The access information will be a controlled set of license options. In the Administration platform, the CP also has the possibility to add group based permission to the whole collection.

- request: persistent identifier, mime-type, access information, location of the digital object (this can be on local disk, URL, or as part of the HTTP post), API-access key, and a content checksum of the original object. The checksum can be used as a mechanism to ensure the correct item is transferred to the SOR. In the Ingest Platform section, this option is further elaborated upon.
- response: XML with status-code

Dissemination API

The dissemination API is the single point of access for all requests for digital objects in the SOR for both human web-users and machine-to-machine interaction. When a request is made to this API with only a persistent identifier the response will be a jump-off page (either as HTML, XML, etc) that contains links to the different available derivatives for the digital object and the license conditions. To some links the description and license information alone will be available, depending on the access restrictions. The links consist of the API base-URL plus the PID, format, size and if applicable the API-access key. The PID refers to the master object that is submitted via the submission API. The derivatives are all linked to the master PID. The sizes and formats of the derivatives are stored as part of the Technical Metadata of the master object identified by the PID.

- request: PID, size, format, API-access key, (output format: JSON, XML, HTML, etc.)
- response: jump-off page when only the PID is given. When the other parameters are given direct access to the object is supplied.

Administration API

The administration API will consist of different components that give access to the different parts of the Administration platform. The rendering layer of the Administration platform will use the same API. A subsection of this API will be made available to partners so they can use this information on their local websites. For authentication a web-services/API key will be made available via the user/role management component.

IAA: Identification, Authentication, Authorisation

The SOR has an identification, authentication and authorisation system. This is necessary to act on access restriction rules, which apply to categories of users in combination with types of usage of digital objects. This feature makes the repository a “trusted repository”: the archival collections entrusted to the CPs are not always publicly accessible due to the privacy of personal papers. The repository should enforce restrictions on access in a very secure way. Digital objects that are publicly available will be made directly accessible via the dissemination API when besides the persistent identifier also the format and size parameters are given. Without the format and size parameters, the jump-off page for the requested object is returned.

The IAA system will support both web-services key (wskey) and user/password based authentication. Based on the access matrix and the access information from the Technical Metadata, the IAA system will determine if and to which formats the requester has access to. The IAA system will authenticate all access to the SOR and will be role-based. The roles will be specified in the Access and Use Specifications and Use-Case documents. The role of the user is also part of the access matrix.

Ingest Platform

The Ingest Platform will validate the submission request from the submission API. The validation might possibly also include virus checking of the digital object. Although for security reasons this check could also be done earlier in the workflow. After validation the ingestion platform adds the request on the processing queues for storage of the object and the technical metadata. The technical metadata will also contain a checksum of the digital master. The digital master is stored with the checksum as the identifier in the Digital Object Repository. This will ensure that no duplicates will be stored in the SOR and that updating the digital master attached to the persistent identifier is a straight forward replacement. It also means that multiple persistent identifiers can refer to the same object stored in the SOR. In addition, the checksum is used to make sure that the item has arrived uncorrupted via the web. It can also be used as a quality check to ensure the object is correctly stored and preserved by the SOR.

Administration platform

The access to the administration API should be handled by the IAA component. The widgets should also be accessible from the institutional websites. Since not all institutional websites will have been built in JAVA, the XML API will have as an additional benefit that it will be easy for partners to create views on their data in the SOR in their preferred technologies.

Some of the envisioned functionality of the Administration platform (note that this list is not exhaustive):

- Overall statistics:
    - Number of data-sets
    - Number of providers
    - Number of digital masters
    - Number of derivatives
    - Access request per data-set, provider, per object
- Monitoring:
    - progress of import process,
    - progress of conversion process,
- Data-set settings:
    - determine access privileges,
    - determine which derivatives are created,
    - determine which derivatives are pushed to social websites
- Role/Group management
    - add users to groups
    - determine group based access privileges

Technical Metadata storage

For the SOR to manage a digital object correctly some basic technical metadata must be supplied during the submission phase. For example, a persistent identifier, mime-type of the object, and license/access information. This information is used by various other components of the SOR to manage the workflow. In addition, a CP provided checksum during submission is also being considered. This checksum will be used for duplicates detection, quality assurance (whilst receiving the object and during storage), and as storage id in the digital object depot.

The technical metadata will be stored in a replicable document database. This database is an integral part of the SOR. Because the Technical Metadata storage must be able to function in a cluster the information must be redundantly available. The ingestion platform is responsible for storing the initial record. Several components can update a technical metadata record:

administration platform, processing queue (i.e. digital object is stored, derivative stored, object flagged for push to social websites, etc.).

Digital Object Depot

The digital object depot is where all the digital masters are stored. The store will have to be replicated to provide redundant storage. The stored digital object is identified by the content checksum. The checksum is stored as part of the technical metadata record for each digital master.

Convert Platform

The Convert Platform should be able to handle a wide variety of formats and create derivatives in most current web-standards. Since the creation of the derivatives - especially from movies and HD master files - is computationally very expensive parallelisation of these processes is an absolute requirement. The convert platform should therefore seamlessly interact with Processing Queue Manager to acquire transformation tasks and be able to run stand-alone on different nodes in the cluster. In order to make the Convert Platform easily extendible, a plugin-based approach for the different converters is the preferred option.

Derivative storage

The derivative storage is responsible for managing the derivatives of the digital master objects that are stored in the Digital Object Depot and are created by the Convert Platform. Although the derivative storage is a separate component in the High Level Design, it is most likely that both the digital master and the derivatives are stored in the same storage mechanism. For the moment, we assume that the SOR will create derivatives for both Video and Image digital masters.

The Derivative Storage should interact with the Cluster Manager. It is likely that the Derivative storage will consist of multiple shards that need to have a single interface to query for and insert derivatives. This multi-node setup of the storage will ensure high throughput for Delivery platform and Convert platform. In a distributed environment the storage and retrieval of derivatives could easily become a bottleneck.

Cluster Manager

The cluster synchronisation/replication manager is responsible for distributing the digital object and technical metadata across the cluster. Although this functionality will most likely be supplied by the technical storage solutions that will be chosen for the SOR, it is still an important part of the SOR. It is a requirement that these storage solutions have some kind of API that makes it possible to integrate information on the state of the cluster in the Administration Platform API.

Processing Queue Manager

The Processing Queue Manager should enable a transparent work-flow between the different components of the SOR. The benefits of an Event Driven Architecture where the components interact with each other through queues is that it becomes much easier to distribute work in the cluster (e.g. use cloud-based solutions to dynamically scale up processing capacity during peak-times) and to use state-based work-flows to prioritise tasks on the queue.

Delivery Platform

An important function of the repository is the interfacing platform responsible for delivering digital objects from the repository upon request (directly to end-users or to external systems). The delivery platform should be capable of accessing derivatives of the master digital copy into a wide variety of formats from the derivative storage. The jump-off page is generated from the Technical Metadata record of the requested PID. It will also need to interact with the IAA to determine if the requested object is available based on the requester's access privileges.

The Delivery platform will be a web application server that will most likely also need to be clustered. Due to the stateless nature of the Dissemination API scalability can easily be achieved through horizontal scaling of the web application servers.

The delivery platform also contains a push service that is able to push derivatives to social sites like Flickr and Youtube. The CP has the possibility to specify in the Administration platform which objects will be pushed to the external services.

Related Components

The related components are part of the SOR ecosphere, but are not considered core-functionality. These components interact with the SOR and provide valuable additions to the work-flow and manageability of the HOPE system.

- **HOPE Persistent Identifier Service:** The Ingest Platform will interact with a number of Persistent Identifier Services, like for example Handle. The main purpose of this interaction will be to make sure that the right information is added to the properties of the PID, so that the ingestion-process requires a minimal amount of additional Technical Metadata. As stated before, only objects with a persistent identifier will be able to be submitted to the SOR. HOPE will also offer its own dedicated Persistent Identifier service to make it easier for CPs to implement PIDs in their system without having to maintain a PID service themselves. The CPs are responsible for requesting an institutional PID, before the HOPE Persistent Identifier Service can host their PIDs for them.
- **3rd party Webstores and reproduction units:** The webstores and reproduction units will interact with the delivery platform via the dissemination API to transparently handle transaction for users that want to buy an object that resides in the SOR. They can make use of web-payment services like Paypal.
- **SOR submission tool:** The SOR submission tool is a small command-line tool that will be supplied to the CP. It will process the XML processing instructions and make the API calls to the SOR ingestion API. The SOR submission tool thus takes care of all the boiler-plate of using the Ingestion API and provides convenient local validation of the correctness of the XML processing instruction file.
- **Staging area (previously known as pre-ingest area):** Since not all content providers are able to store all digital objects online or send them via http, a scenario with FTP upload should be supported as well. The basic idea is that the CP uploads the objects to the staging area together with an XML processing instruction. This instruction contains all the parameters to construct calls to the Submission API. From the Administration platform, the CP should be able to trigger a run of the importer that reads the processing instruction and turns them into Submission API calls. The CP should be able to track the progress of the import via one of the Administration Platform widgets.
  The benefit of creating a staging area early on in the project, is that the CPs can start processing their collections over a long period of time as opposed to creating bottlenecks before go-live time.

Conclusion

In this document we have described the High Level design and work-flow of the HOPE system. We have focused on providing a transparent discovery-to-delivery process for the users and simple maintainable work-flow for the HOPE content providers. The system is designed with future maintainability, sustainability and scalability in mind. By the completion of the project the system should be operational and should be able to sustain itself afterwards through the various cost-recovery strategies developed by the HOPE Business Plan in WP7.

Appendix: Sample XML Processing Instruction

Below you find the first draft for the processing instruction that the HOPE SOR submission tool needs to submit each object to the SOR. The SOR submission tool will be a small standalone java tool that will be supplied by the HOPE to the Content Providers to simplify the submission to the SOR.

```
<hope_sor api_key=”123/456”>
        <object action=”add”> <!-- default: add but delete other option -->
                <pid>..../1066/1</pid> <!-- persistent identifier -->
                <mime-type>image/jpeg</mime-type> <!-- standard http mime-types -->
                <access>free</access>
                <location>/Volumes/hope/coll1/image.jpg</location>
                <checksum>42eac3764dea6aabf4c92add6beb636a</checksum>
        </object>
        ..... <!-- more objects -->
</hope_sor>
```