Deliverable D5.1

HOPE
Grant agreement no: 250549
Heritage of the People's Europe

Repository Infrastructure and Detailed Design

•Deliverable number:	D5.1
•Status:	FINAL
•Authors:	Jerry de Vries
•Delivery Date:	01-04-2011
•Dissemination level:	Public

HOPE is co-funded by the European Union through the ICT Policy Support Programme.

Version history

Date	Changes	Version	Name
25-02-2011	First draft	0.1	Jerry de Vries
01-03-2011	Schemas added	0.2	Jerry de Vries
02-03-2011	Technical description of components added	0.3	Jerry de Vries
09-03-2011	UML diagrams added, design choices updated	0.4	Jerry de Vries
11-03-2011	Updated design choices	0.5	Jerry de Vries
14-03-2011	Added conclusion	0.6	Jerry de Vries
24-03-2011	Changes made based on the first reviews	0.7	Jerry de Vries
25-03-2011	Changes made based on the last reviews. Updated appendix in separate document.	0.8	Jerry de Vries
25-03-2011	Added PPSS as tool and last check up	1.0	Jerry de Vries
28-03-2011	Described PID service in separate chapter	1.1	Jerry de Vries

Contributors

Institution	Name
IISG	Gordan Cupac
	Mario Mieldijk
	Sjoerd Siebinga
	Titia van der Werf
	Lucien van Wouw
CNR-ISTI	Alessia Bardi
	Paolo Manghi
	Franco Zoppi

Table of contents

HOPE is co-funded by the European Union through the ICT Policy Support Programme.

Introduction

The HOPE system consists of different parts. These parts are the local systems of Content Providers, the HOPE Aggregator, the HOPE PID service, the HOPE Shared Object repository (henceforth SOR) and the discovery services.
Figure 1 shows a diagram of the component parts of the HOPE system and of the data-flows can be found. This diagram is derived from the high level design[1].
Figure 1 shows a proposed updated version of the diagram. In the hope consortium is agreed that the HOPE SOR won't provide the upload to social sites. Therefore it is left out and not mentioned further in this document.

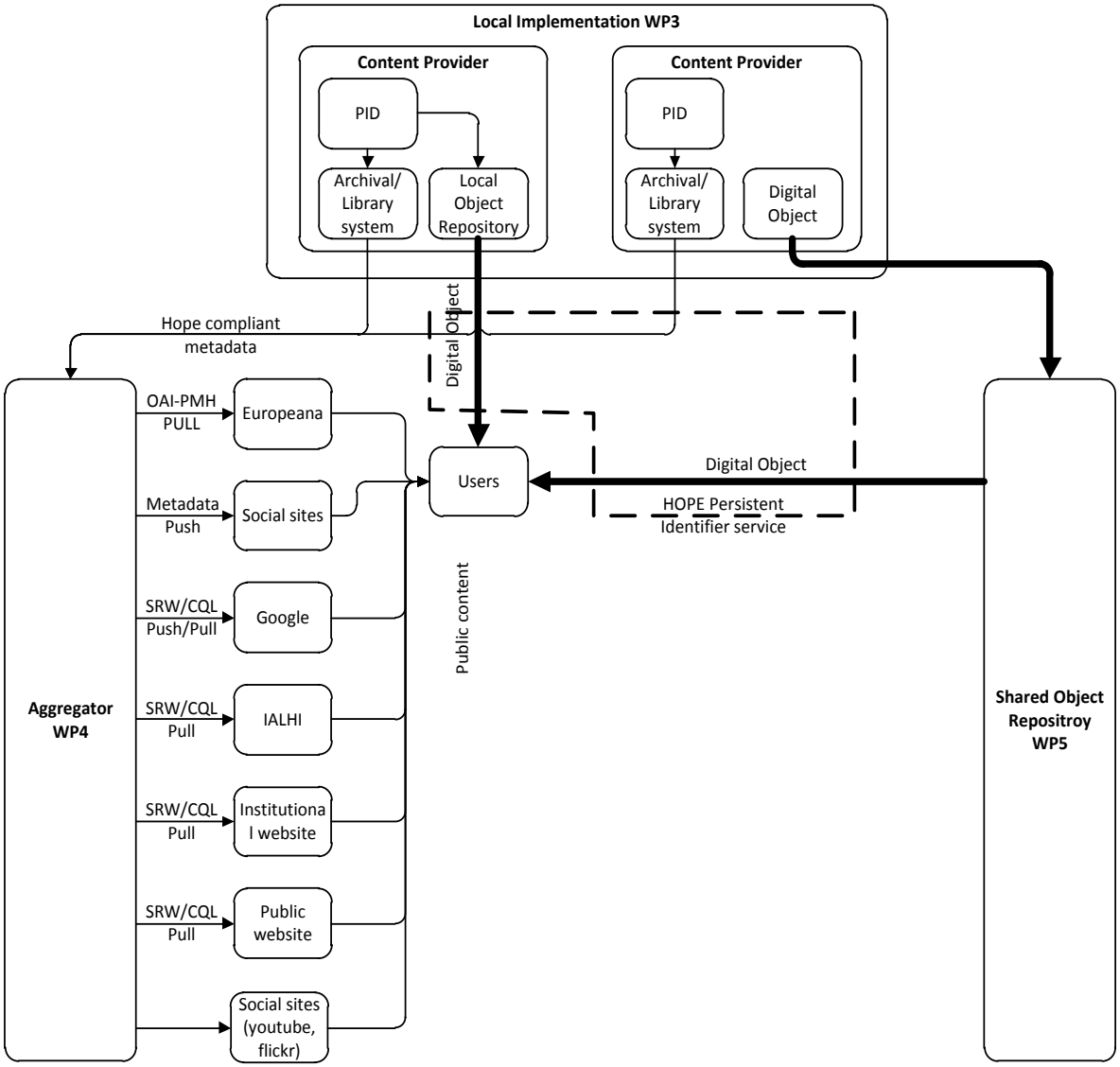


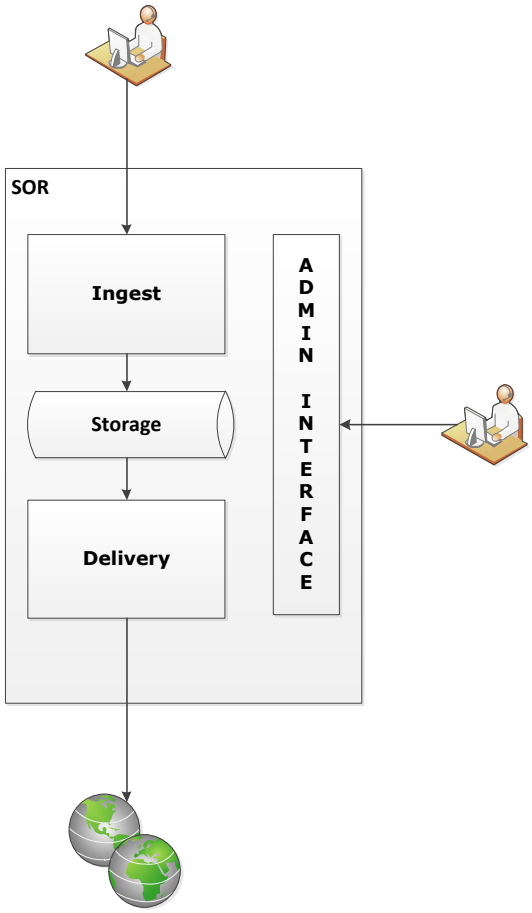
Figure 1 High level design diagram

[1] See T2.1 HighLevelDesign v0.1

This document defines the detailed design, infrastructure and technical architecture of the Shared Object Repository (SOR). The input for this document comes from: The High Level Design WP2 (T2.1), gathered requirements from the Content Providers (henceforth CP) in the "HOPE consortium" and the milestone 5.1 document[2]. This document also contains the design and requirements of the HOPE Persistent Identifier (PID) service.

Requirements SOR system

Derived from the Milestone 5.1 document[2] we can see that the SOR basically consists of three parts: 1) Ingest (which is also storage), 2) Delivery and 3) Administration interface. Figure 2 shows a diagrammatic representation of the SOR.



Before the discovery to delivery process (d2d) can take place, digital objects should be ingested into the SOR.

As digital masters are usually large files, they are not fit for large scale online delivery via the web, so by default they have a restricted access status and the SOR creates smaller size derivatives out of them, for delivery. It is the Content Provider (CP) who sets the policies and rules for access to the digital object and its derivatives.

To see how the three basic processes of the SOR can work, we have to describe the SOR and the components of the SOR in more detail. This document zooms in on the SOR and describes all of its components and infrastructure of these components.

Figure 2 SOR basic

[2] Milestone document M5.1 - Repository workflow and Requirements specification

Requirements from the High Level Design

- Use of Persistent Identifier System
- Scalable for > 500Tbytes
- Scalability for Performance (down- or up scaling)
- High availability
- Cost-effective
- Low Maintenance
- Object oriented architecture
- Simple, clean and open design
- Must be extendable for future extensions (preservation, multiple copies, caching derivatives)
- Easy to manage
- It is preferable that the content providers can easily setup there local SOR with the components that are used in de SOR
- All software must be distributable
- Safe (secure) storage

Requirements from the Content Providers

- All the requirements and specification for the SOR are collected and updated in the Milestone document **M5.1 - Repository workflow and Requirements specification**

Chapter overview

Chapter 1: Describes the high level design of the SOR. In chapter 1.1 gives an explanation of each component of the SOR.
Chapter 2: Describes the High Level and Low Level design of the PID service
Chapter 3: Describes the low level design of the SOR. Chapter 3.1 describes the infrastructure between the components of the SOR. Chapter 3.2 describes the tools and software that will be used to implement the components of the SOR. In chapter 3.3 the design choices are highlighted. Chapter 3.4 describes the technical implementation and chapter 3.5 describes the low level design dependencies.

1. SOR Detailed design

This section describes the detailed design for the SOR. 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. In Figure 3 a diagrammatic representation of the Shared Object Repository can be found.

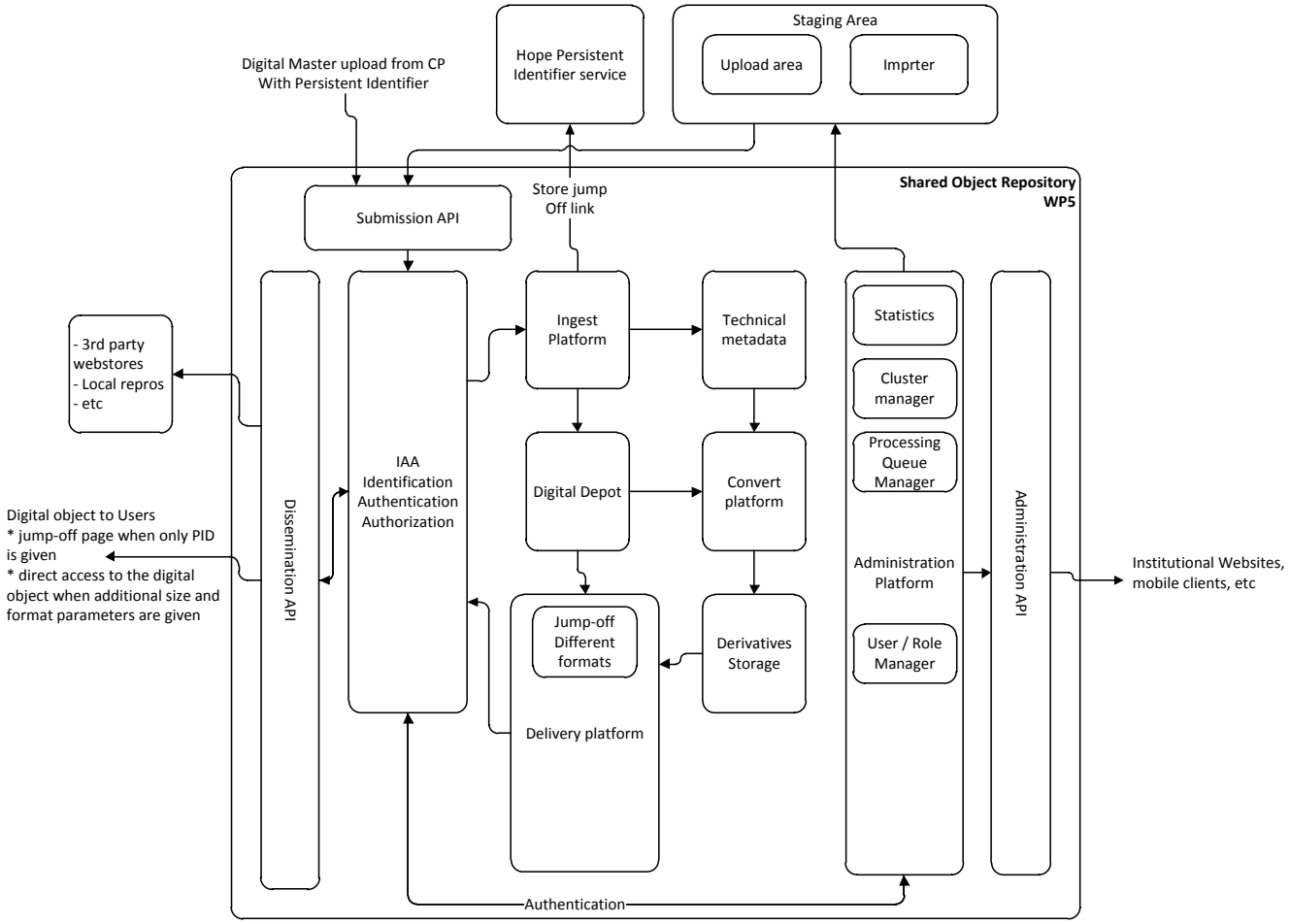


Figure 3 SOR detailed design

Figure 3 shows the components of the SOR. The diagram also shows the communication between the components. The following chapter describes all these components in detail.

1.1 SOR components

This chapter gives an overview of all the components of the SOR. A description of the function is given and the technical details of each component is given

1.1.1 Submission API

The submission API is responsible for receiving a submission request for storing a digital master in the SOR. The SOR processing instruction also contains an option to send a delete or update request for the digital master. The access information will be controlled by the access rights (open or restricted access, for more details see HOPE access conditions matrix).

1.1.2 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 an http request is made to this API with the PID of the digital object, the response will be a jump-off page (either as HTML, XML, etc) that contains links to the master file and the different available derivatives for the digital object. The links that are shown on the pages are based on the access rights of the digital master. When the access is open all links will be shown. When access is restricted the link to the master file won't be shown at the jump-off page. The PID refers to the master file 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 file identified by the PID. These derivatives are accessible by providing a parameter extension to the PID. This parameter indicates which derivative level is requested.

1.1.3 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. For authentication a web-services/API key will be made available via the user/role management component.

1.1.4 IAA: Identification, Authentication, Authorization

The SOR has an identification, authentication and authorization system. This is necessary to act on access rights 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 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. The IAA system will support both web-services key (wskey) and user/password based authentication. Based on the HOPE access conditions 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-base.

1.1.5 Ingest platform

The Ingest Platform will validate the submission request from the submission API. The validation also includes virus checking of the digital object. 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. In addition, the checksum is used to make sure that the item has arrived uncorrupted via the web. It will also be used as an integrity check when storing and preserving the object in the SOR.

1.1.6 Administration platform

The access to the administration API will be handled by the IAA component. (See Milestone 5.1 document[2] for more details). The platform gives a status overview to the Content Provider (henceforth CP). The CP is able to: 1) view his collection of objects, i.e. how many objects are stored in the SOR and how many objects are ready for submission. 2) retrieve a status overview of the ongoing submission process and 3) usage statistics. The CP can manage and carry out submissions from this platform.

1.1.7 Convert platform

The Convert Platform handles a wide variety of formats and creates derivatives in most current web-standards. The convert platform interacts with the Processing Queue Manager to acquire transformation tasks and be able to run stand-alone on different nodes in the cluster.

1.1.8 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 is capable of accessing derivatives

of the master digital copy into a wide variety of formats (See Milestone 5.1 document[2] for supported 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.

1.1.9 Technical Metadata storage

For the SOR to manage a digital object correctly some basic technical metadata must be supplied during the submission phase; an API key, resolver URL, naming authority, access rights, Local Identifier/PID, action, location, checksum, mimetype. This information is used by various other components of the SOR to manage the workflow. A CP can provide a checksum during submission or a CP can allow the SOR to generate a checksum. 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. 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. Several components can update a technical metadata record: administration platform, processing queue.

1.1.10 Digital Object Depot

The digital object depot is where all the digital masters are stored. The store will 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.

1.1.11 Derivative storage

The derivative storage is responsible for managing the derivatives of the digital master files that are stored in the Digital Object Depot and are created by the Convert Platform. The SOR will create derivatives for both Video and Image digital masters. The Derivative Storage interacts with the Cluster Manager. The Derivative storage 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.

1.1.12 Cluster manager

The cluster synchronization/replication manager is responsible for distributing the digital object and technical metadata across the cluster. These storage solutions have an API that make it possible to integrate information on the state of the cluster in the Administration Platform API.

1.1.13 Processing Queue Manager

The Processing Queue Manager manages the 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 are that it becomes much easier to distribute the 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 prioritize tasks on the queue.

1.1.14 Staging Area

Since not all content providers are able to store even temporarily, large collections of digital objects online, a staging area with SFTP upload is provided. The CP uploads the objects to the staging area together with the SOR processing instruction. This instruction contains all the parameters to construct calls to the Submission API. From the Administration platform, the CP is able to trigger a run of the importer that reads the SOR processing instruction and turns them into Submission API calls. The CP can track the progress of the import via the Administration Platform.

2. Persistent Identifier Service

As the Persistent Identifier Service (PID service) is not an actual part of the SOR, the PID service is described separately in this chapter.

2.1 High Level Design PID Service

The HOPE Persistent Identifier Service is a separate service related to the HOPE system (See Milestone 5.1 document[2] for more details.).

The HOPE persistent Identifier Service is an implementation of a Handle[3] webserver. Through a soap protocol other web services can interact with this web service. At the time of writing a pilot web service is accessible via the following URL: http://195.169.122.195/pidservice/handle.wsdl[4]

This URL describes the interface of the web service. An example of this interface is shown in Appendix A.

2.2 Low Level Design PID Service

Based on the design choices, described in chapter 3.3 the PID service will be implemented as follows:

PID Server

ServerName: victoradler<following number>.objectrepository.eu
Role(s) High Level Design: HOPE Persistent Identifier Service
Technical Specs:

- Xen virtual server
- 1 vCPU
- 512 MB memory
- 5GB vDISK

Responsible for:

- Delivering Persistent Identifiers for 'HOPE' metadata and objects if the Content Provider cannot supply the PIDs

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- Apache Tomcat

[3] http://www.handle.net/
[4] http://www.w3.org/TR/wsdl

3. Low level design

3.1 Infrastructure

In milestone 5.1 document[2] are the workflows described for the SOR. For **release 1** of the SOR the following infrastructure is created. The infrastructure is presented in the following diagrams

The basis is as follows:

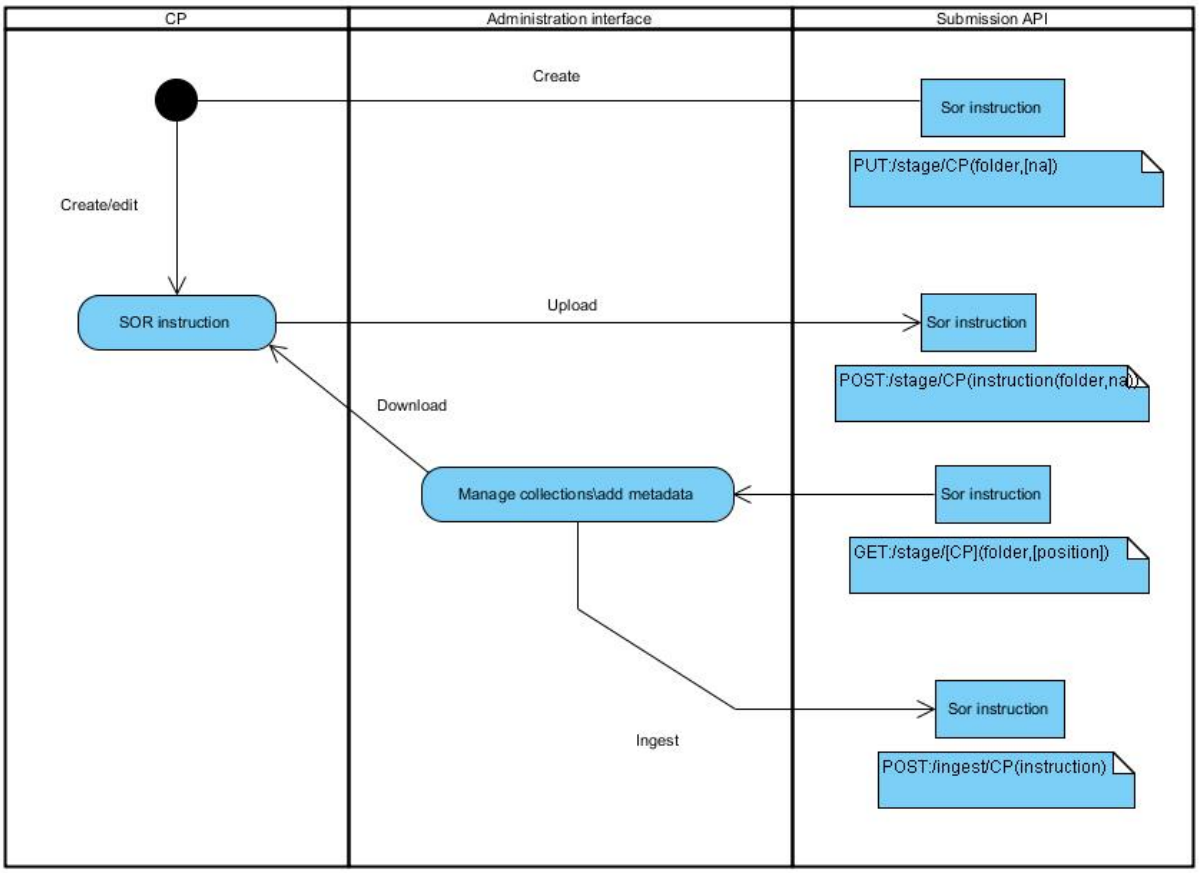


Figure 4 Creating and managing SOR processing instruction

A CP has to create a SOR processing instruction. A CP can create one manually, or the CP can instruct the SOR to create one. If the CP has created the SOR processing instruction manually, the CP has to upload the SOR processing instruction to the SOR.
On the administration panel the CP is able to manage the SOR processing instruction and add metadata. From here the CP can start an ingest or the CP can download the SOR processing instruction for editing.

The creation of the SOR processing instruction is the first step in the process.

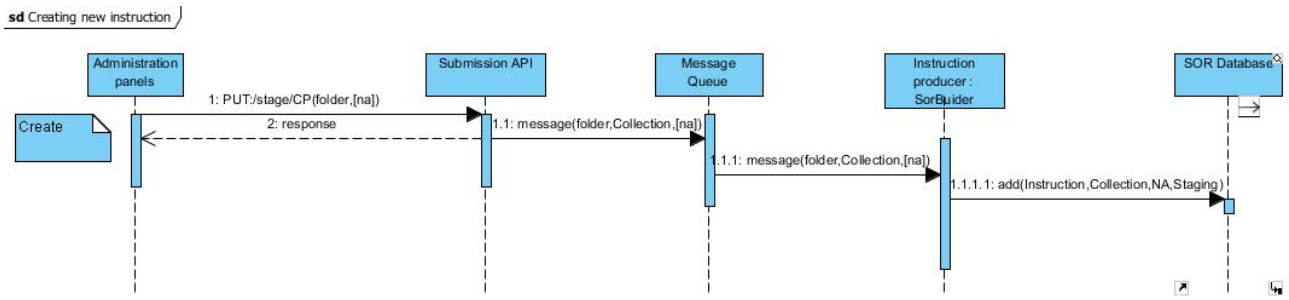


Figure 5 User creating SOR processing instruction

On the administration panel the CP selects the option to generate the SOR processing instruction. The administration platform calls the submission API. The submission API put the request on the processing message queue.

When the SOR processing instruction producer receives the request to build the instructions, the actual SOR processing instruction will be build.

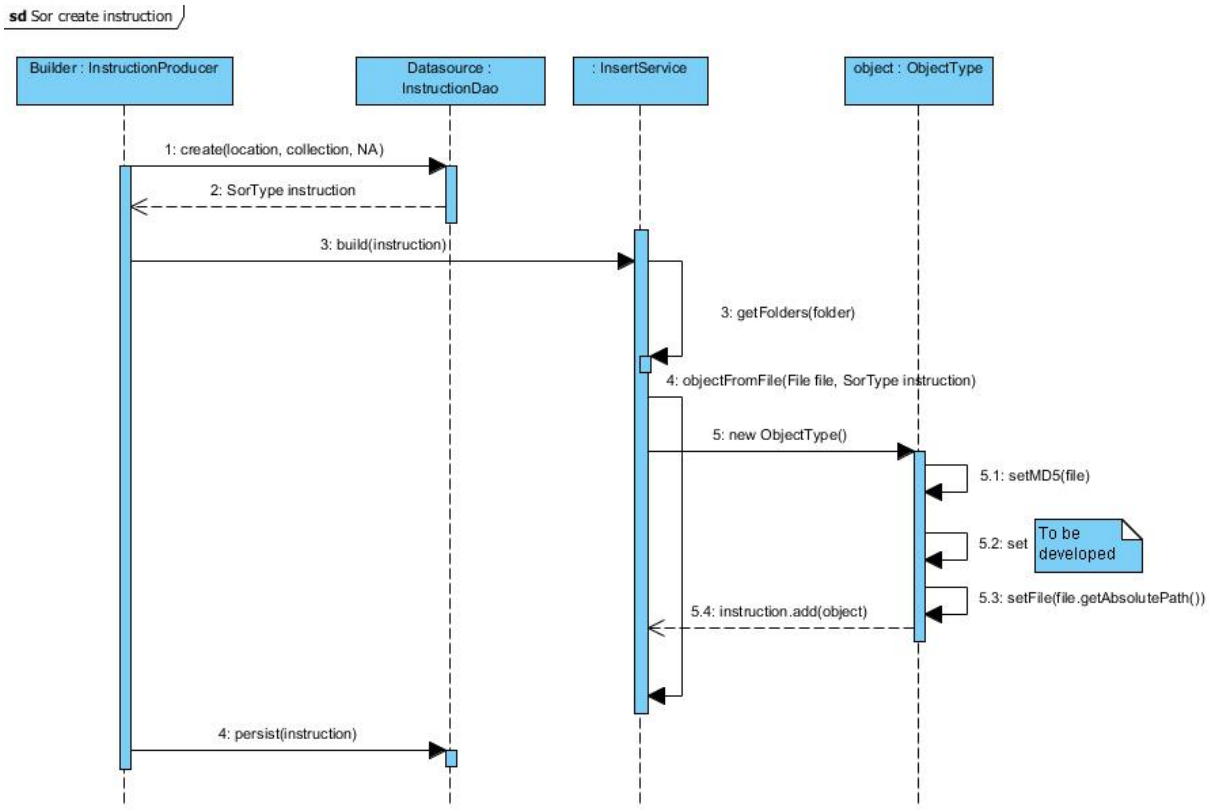


Figure 6 SOR creating SOR processing instruction

The builder starts the build. The builder retrieves the root folder which contains the files. For each file the instruction will be created and added to the SOR processing instruction. If all file are present, the SOR processing instruction will be returned.

When a SOR processing instruction is available, the CP is able to update the SOR processing instruction.

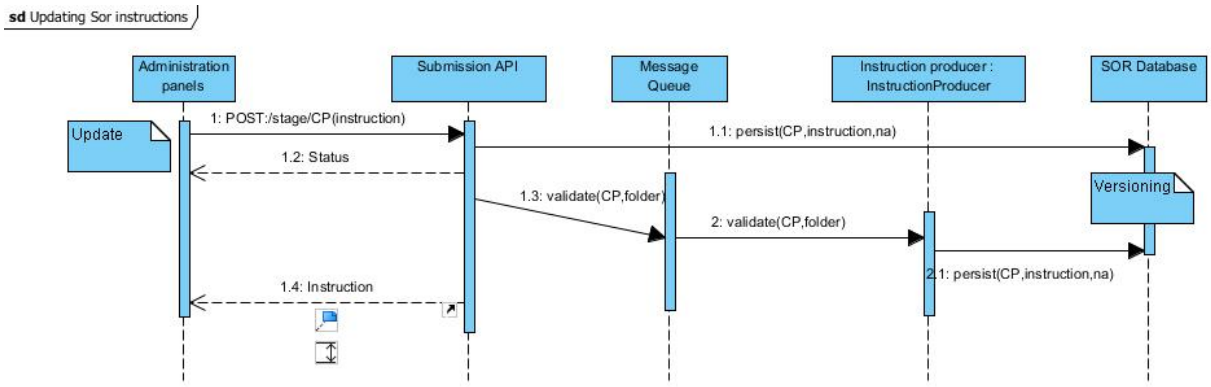


Figure 7 Updating SOR processing instruction

From the administration panel the CP selects the update. A message to the submission API is sent which will ask the status of the previous SOR processing instruction.

At ingest the SOR processing instruction should be retrieved from the SOR database.

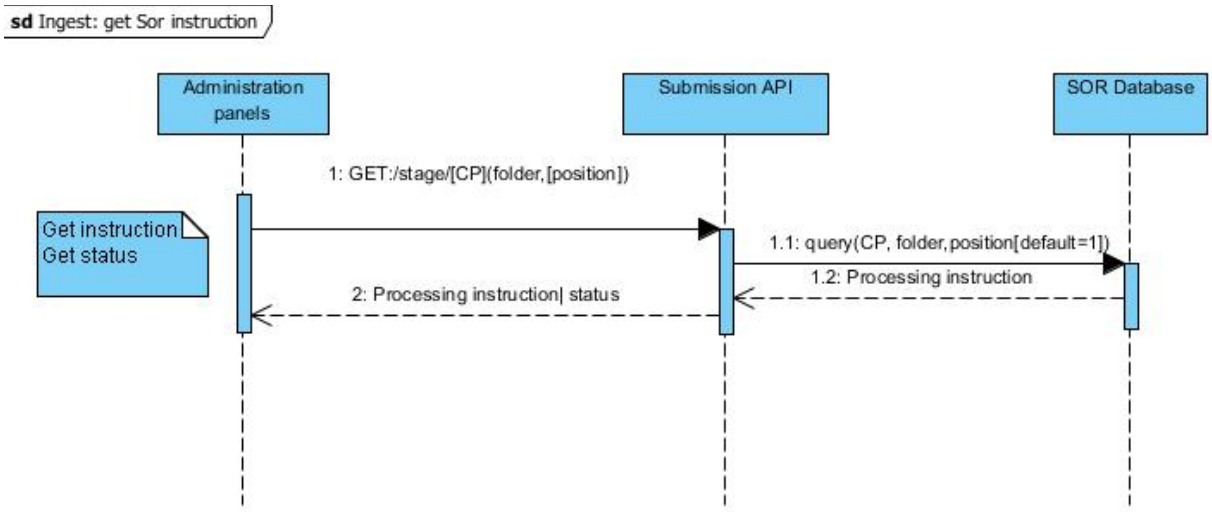


Figure 8 Get SOR processing instruction for ingest

A get request will be sent to the submission API, which will retrieve the SOR processing instruction from the SOR database. The SOR processing instruction will be returned or a status will be returned.

When the SOR processing instruction is retrieved the actual ingest can take place. The actual ingest can take place in release 2. The processing instruction will be executed accordingly, where each file will be ingested into the Depot

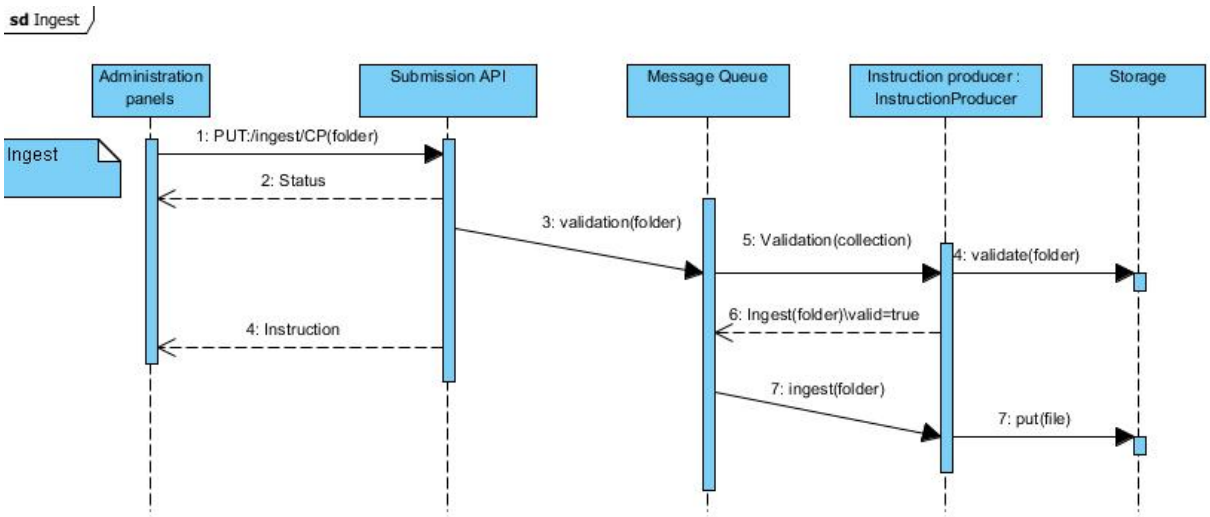


Figure 8 ingest

HOPE is co-funded by the European Union through the ICT Policy Support Programme.

3.2 Tools and software

In chapter 1.3 we can see the infrastructure for the first release of the SOR. These components will be developed. For some components existing tools will be used. In this way the infrastructure is completed. This chapter describes all these tools that will be used.

During the project this chapter will be updated. During each release new components will be implemented. For each release this chapter will be updated with the specification and requirements for the components that will be implemented.

3.2.1 Software

Drupal

During the first release the implementation of the administration platform will start. This platform will be implemented in Drupal[5]. Drupal is a free and open source content management system, which is extendable with different modules. Drupal provides a powerful user and role management. Adding content dynamically is easy and therefore Drupal is suitable to provide statistics and status updates of the SOR automatically.

For the implementation of the administration panel the latest version of Drupal will be used; version 1.7.0

3.2.2 Tools

During the first release of the convert platform will be implemented. The focus is on converting TIFF to JPEG and resize of TIFF files (i.e. creation of derivative level 3, 200px.). Based on the following survey ImageMagick[6] seems to be the most suitable tool as ImageMagick fits the requirements the best (see Milestone 5.1 document[2]). This tool will be proofed during the first release.

ImageMagick

ImageMagick is the most used image processing program online. ImageMagick is used to create, edit, and compose bitmap images. It can read, convert and write images in 120+ formats including TIFF, JPEG, JPEG-2000 and PNG. You can use ImageMagick to translate, flip, mirror, rotate, scale, shear and transform images, adjust image colors, apply various special effects, or draw text, lines, polygons and ellipses.
The functionality of ImageMagick is typically utilized from the command line or you can use the features from programs written in your favorite programming language.

[5] www.drupal.org
[6] http://www.imagemagick.org/

ImageMagick is free software delivered as a ready-to-run binary distribution or as source code that you may freely use, copy, modify, and distribute in both open and proprietary applications. It is distributed under an Apache 2.0-style license, approved by the OSI and recommended for use by the OSSCC.

For the implementation of the converter platform the latest 64 bits version 6.6.8-6 of ImageMagick will be used.

PPSS

PPSS is a Bash shell script that executes commands, scripts or programs in parallel. It is designed to make full use of current multi-core CPUs. It will detect the number of available CPUs and start a separate job for each CPU core. It will also use hyper threading by default. PPSS can be run on multiple hosts, processing a single group of items, like a cluster.

You can provide PPSS with a source of items (a directory with files, for example) and a command that must be applied to these items. PPSS will take a list of items as input. Items can be files within a directory or entries in a text file. PPSS executes a user-specified command for each item in this list. The item is supplied as an argument to this command. At any point in time, there are never more items processed in parallel as there are cores available.

From version 2.0 and onward, PPSS supports distributed computing. With this version, it is possible to run PPSS on multiple host that each process a part of the same queue of items. Nodes communicate with each other through a single SSH server.

For the implementation of the converter platform the latest version 2.85 of PPSS will be used

- **Remark:** If updates or newer versions will be published for the above tools and software, these will be implemented. This document will be updated instantly.

The low level design is based on the high level design. In this chapter all servers are described in detail, including the dependencies of the low level design. The diagrams with the overview of the low level design are shown in Appendix B.

3.3 Design Choices

At the start of the project we first carried out a review of existing 'repository software'. Although some software like Fedora-commons, E-prints and others where good candidates, the requirements could not be fulfilled by these software solutions. The reasons not to choice for this software are for example:

- The software is not build up modular, one-package is used for the entire system/application (blackbox).
- Some software uses clients (not browsers)
- Not so easy scalable for performance or storage growth and future extension
- The software highly depends on SQL servers, which is a RDBMS solution. Therefore it is less suitable for large file.

As known in software development 'object oriented architecture' is often be used to meet scalability and flexibility requirements. That's why we have made the decision to pull this idea to an 'operating system level', so that we can meet more requirements at once. With the use of Virtualization technologies we can put almost every software component in the SOR on one server, without the costs of more physical servers (hardware). However we shall implement two physical servers for I/O consuming software like the Postgres-sql server.

3.3.1 Technical solutions

One of the design choices is to put each component on its own virtual (web)server. In this way we get a modular infrastructure and it is easy to implement new components to the SOR, in such a way it is easy to change a component with an updated version of the component.

3.3.1.1 Virtualization

The keyword for meeting many of the technical expectations in the design of the SOR is Virtualization. Virtualization solves requirements like High-availability (hardware level), keeping the costs of the whole system low and the long term infrastructure (power and rackspace) manageable.
In our situation we have chosen for de Citrix XEN virtualization solution. The reasons why:

- It is proven technology

- It is cost effective (available at no costs)
- If needed you can add more features for a per machine license (not like other virtualizations products)
- XEN is also available in other Linux Distributions (CentOs, Red-Hat, Ubuntu)
- Citrix delivers a free management tool for all XEN servers (XenCenter)
- Easy to implement, well documented on the Internet
- Easy to backup whole virtual machines with the build-in tools

Meets the requirements: High Availability, Cost effective, Scalability for Performance, Easy to manage, "Simple, Clean and Open design"

3.3.1.2 Storage

We have decided that the storage solution must be simple, cost effective and stable. So we have chosen for the storage independent software solution called MongoDB. MongoDB provides a software storage layer on top of the hardware storage layer. This gives us (or Local SOR) the possibility to choose a hardware vendor and/or hardware solution (like nfs, iscsi san, fc san) that fits our needs. The configuration options within MongoDB take care of the High –Availability/ multiple copies/sharding/ and so on. In our case we have chosen for the two Dell MD3000i SAN with redundant controllers with two extra Dell MD1000 diskcabinets. The Dell MD3000i SAN is a cost affordable product that has proven to be stable and has good support options by the hardware vendor.

Meets the requirements: Save (secure) storage Scalable for > 500Tbytes, Scalability for Performance (down- or up scaling), High availability, Cost-effective, Low Maintenance

3.3.1.3 Network

Core Switch

To connect all the network components (network interface cards, firewalls, routers) together we have decided to put a redundant switch solution from Dell as core-switch. Dell delivers with the Power-Connect 6224 series a cost effective, high availability solution for the SOR. The switches will use VLAN's to separate the different kind of networks, like ISCSI, DMZ, WAN, Server LAN, Dev LAN.

Switch

To connect the network components from the Datacenter Vancis[7] to the Datacenter IISG, we have decided to use one Cisco 2960 switch on each side. We didn't choose Dell switches, because our network provider only allows us the use of VLAN Trunking if Cisco is used. If we take a look at the LLD-specified (See Appendix B) this looks like a single point of failure, but less is true. The solution on the Datacenter Vancis side is all redundant to the world, but the switch to the remote copy in the Datacenter-IISG is not. If a switch failure will exist between the datacenters, and it is resolved, the software will take care of copying the new information/data to the remote Datacenter, in our case from Datacenter Vancis to Datacenter IISG (remote copy). Although the risk is very low there is a possibility to extend the number of switches.

Firewalls

To add security to the whole SOR system, we have decided to use the redundant firewall solution from Fortinet. These firewalls are provided by the IISG and are well featured. One of the interesting features is to add more Virtual Firewalls, which gives the ability to delegate a firewall solution for a particular network (like HOPE). Also the use of multiple WAN ports is supported which gives us scalability in download and upload bandwidth.

Internet

The redundant Internet connectivity is provided by the KNAW and has a bandwidth of 1Gb/s full duplex. The KNAW is connected to SURFnet with a redundant connection of 1 Gb/s.
In April 2011 the KNAW connectivity to SURFnet will be upgraded to 10Gb/s but limited to 3Gb/s. The Internet connectivity between the firewalls of the SOR and the KNAW will be upgraded to at least 2Gb/s.

Meets the requirements: Scalability for Performance (down- or up scaling), High availability, Cost-effective, Low Maintenance, Easy to manage

3.3.1.4 Software

Operating system

Because the aggregator already uses Ubuntu Linux and to keep the whole 'HOPE system' simple, clean and open, we have decided to use Ubuntu Server Linux LTS 10.04.x 64 bits in the SOR. Ubuntu is a widely used Linux distribution with a big software repository and if needed you can add/buy additional support from Canonical (the company behind Ubuntu). Ubuntu Server Linux Long-term support

[7] See appendix C for an explanation of KNAW, SURFnet, Vancis and IISG

(LTS) releases are especially designed for stable deployment scenarios and are supported for five years.
The only commercial (academic licensed) software in the SOR are two Microsoft Windows 2008 R2 standard servers. These are necessary for XenCenter to distributed, roles and users into the XenServers. In a local SOR this is not really necessary, because there are less system administrators, but in our case we need to.

3.3.1.5 Converter

The making of derivatives from 'master files' is a CPU consuming task. We have decided to use a 'blade center' with ten 'dual quad cores CPU' blade servers (provided by the IISG). These servers will be installed with Citrix XenServer, this gives a little bit of overhead but will be compensated by easy management, resource monitoring and high-availability.
The blade servers will be configured in a nine production plus one spare configuration for extra hardware High-Availability. On top of the XenServers we are running two groups of virtual converter servers, one group for the images and one group for the movies/sound. We are using always the same amount of virtual converter servers which gives an easy configuration model. On the fly scaling for faster/slower processing of each group of files is possible to place the right amount of virtual converter servers on less or more blade servers.
Another technique that will be used is parallel processing software, named PPSS (Parallel Processing Shell Script). This technique gives the ability to deliver fast on-demand services when this is requested. This will also be used to handle the bulk of 'master files' that will put into the SOR in initial loading.

Meets the requirements: Derivatives, High-Availability, Scalability for Performance (down- or up scaling), Simple, clean and open design, Object oriented architecture.

3.3.1.6 Management, maintenance and monitoring

To keep everything manageable and maintainable we implement some Open-Source/Free tools into the SOR, they are:

- **XenCenter** for managing all virtual servers from one Interface. It has the ability to make snapshots (backup in time) from virtual servers, make backups off the whole virtual server, move virtual machines from one XenServer to another XenServer (patching without downtime), monitors the virtual server on CPU/Memory/Disk space.
- **Webmin** for managing all virtual servers with a web interface. It has the ability to send 'linux commands', managing local users/groups, monitoring software, install software and much more from one single point.

- **Zenoss** for monitoring all the servers with snmp from one web interface dashboard.
- **Puppet** for patching or change configuration files on all virtual servers from one single point. Makes management and easy roll-out for installation possible.

Meets the requirements: Low Maintenance, Cost-effective, Easy to manage

3.4 Implementation

Based on the design choices the following implementation is designed. This chapter describes this implementation. Appendix B shows a schematic overview of this implementation.

3.4.1 API Servers

API Server

ServerName: hugohaase<following number>.objectrepository.eu
Role(s) High Level Design: Submission API, Dissemination API, Administration API, Delivery Platform, Ingest Platform
Technical Specs:

- Xen virtual server
- vCPU
- 1024 MB memory
- 5GB vDISK

Responsible for:

- Delivering APIs for internal and external services

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- Apache Tomcat

Proxy Server

ServerName: leontrosky<following number>.objectrepository.eu
Role(s) High Level Design: Dissemination API, Jump-Off page, Administration API
Technical Specs:

- Xen virtual server
- 1 vCPU
- 512 MB memory
- 5GB vDISK

Responsible for:

- Handle all the (secure) webtraffic for the backend servers;
  - Jump-off pages
  - Administration interface of the SOR
  - IALHI portal
- Logging visits per virtual hosts
- Deliver een webbased overview of the visits per virtual hosts

Used software:

- Ubuntu LTS 10.04.x 64bit
- Apache webserver
- AwStats
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- Heartbeat (fail over)

3.4.2 IAA: Identification, Authentication, Authorization servers

Directory Server

ServerName: robertgrimm<following number>.objectrepository.eu
Role(s) High Level Design: Identification, Authorization, Authentication (IAA), User/role manager, HOPE access conditions Matrix
Technical Specs:

- Xen virtual server
- 1 vCPU
- 512 MB memory
- 5 GB vDISK

Responsible for:

- IAA, user roles and user authentication for servers

Used software:

- Ubuntu LTS 10.04.x 64bit
- Apache webserver
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- OpenLdap
- PHPldapAdmin

3.4.3 Platform servers

Web Server

ServerName: karlrenner<following number>.objectrepository.eu
Role(s) High Level Design: Jump-Off page, Administration Interface, Helpdesk (future), Tracking system (future)
Technical Specs:

- Xen virtual server
- 1 vCPU
- 512 MB memory
- 5GB vDISK

Responsible for:

- Delivering webpages for the
  - Jump-off pages
  - Administration interface of the SOR
  - IALHI portal

Used software:

- Ubuntu LTS 10.04.x 64bit
- Apache webserver
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- ProFtp
- Drupal 7
- PHP

Converter manager Server

ServerName: augustbebel<following number>.objectrepository.eu
Role(s) High Level Design: Convert Platform
Technical Specs:

- Xen virtual server
- vCPU
- 1 GB memory
- 5GB vDISK

Responsible for:

- Distrubuting the derivative converting job to multiple Converter Servers

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- Parallel Processing Shell Script (PPSS)

Converter Server

ServerName: augustbebel<following number>.objectrepository.eu
Role(s) High Level Design: Convert Platform
Technical Specs:

- Xen virtual server
- 8 vCPU
- 4 MB memory
- 5GB vDISK

Responsible for:

- Converting Digital objects (Masters) to derivatives

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- Parallel Processing Shell Script (PPSS)

Processing Message Broker Server

ServerName: julesguesde<following number>.objectrepository.eu
Role(s) High Level Design: Processing Queue Manager
Technical Specs:

- Xen virtual server
- vCPU
- 1024 MB memory
- 5GB vDISK

Responsible for:

- Managing Converter Processing from Staging Area to Derivative Storage

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- Apache Tomcat

3.4.4 Storage

Database / NFS Server

ServerName: rosaluxemburg<following number>.objectrepository.eu
Role(s) High Level Design: Technical Metadata, Delivery platform, Administration platform
Technical Specs:

- Server
- 1 QC CPU
- 8GB memory
- 250GB DISK RAID 1

Responsible for:

- technical metadata (DB), webdata (DB), fileservice

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- Postgres SQL
- NFS
- DRbd
- Heartbeat

MongoDB Server

ServerName: clarazetkin<following number>.objectrepository.eu
Role(s) High Level Design: Digital Object Depot, Derivative Storage
Technical Specs:

- Xen virtual server
- vCPU
- 2048 MB memory
- 5GB vDISK
- x 3 x 20TB ISCSI DISKS RAID6

Responsible for:

- Delivering APIs for internal and external services

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon

Remark: MongoDB server must run in a cluster (Replicaset) for fail-over.

SAN

ServerName: vlademimrlenin<following number>.objectrepository.eu
Role(s) High Level Design: Object repository, Derivative storage
Technical Specs:

- ISCSI SAN device with Dual Raid controllers
- Multiple CPU's
- Multiple cabinets
- x 3 x 22TB DISKSPACE (Netto)

Responsible for:

- Storing all virtuall machines
- Storing Derivatives from objects (masters)
- Storing objects (masters)

Used software:

- Dell MD storage manager

3.4.5 Staging Area

Staging Area Server

ServerName: victoradler<following number>.objectrepository.eu
Role(s) High Level Design: Staging Area, Upload area, Importer
Technical Specs:

- Xen virtual server
- 1 vCPU
- 1024 MB memory
- 5GB vDISK
- 7 TB ISCSI DISKS RAID6

Responsible for:

- Temporary store the digital object before submission
- Deliver SFTP service

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon

3.5 Low level design dependencies

For the servers described in the previous chapter to operate there are some technical dependencies. These dependencies are described here:

3.5.1 Virtual Servers

3.5.1.1 Applications Environment

Citrix XenServer

ServerName: antonpannekoek<following number>.objectrepository.eu
Role(s) High Level Design: Not named but is nessecary
Technical Specs:

- Server
- 1 QC CPU
- 24GB memory
- 146GB DISK RAID 1
- x 2TB ISCSI DISK RAID1

Responsible for:

- Hosting all virtual servers

Used software:

- Citrix XenServer 5.6.xxx

XenCenter Server

ServerName: karlmarx<following number>.objectrepository.eu
Role(s) High Level Design: not named but is necessary for managing Virtualization Environment
Technical Specs:

- Xen virtual server
- 1 vCPU
- 1500 MB memory
- 40 GB vDISK

Responsible for:

- User authentication for Xen Servers (hosts), active directory, monitoring

Used software:

- Windows 2008 Standard 64bit
- RDP
- XenCenter
- Putty
- Spiceworks (management tool)

Name Server

ServerName: pavelaxelrod<following number>.objectrepository.eu
Role(s) High Level Design: not named but is necessary
Technical Specs:

- Xen virtual server
- 1 vCPU
- 256 MB memory
- 5GB vDISK

Responsible for:

- IPv4 and IPv6 Name resolution services

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- Bind9 nameserver

Mailrelay Server

ServerName: julliusmartov<following number>.objectrepository.eu
Role(s) High Level Design: not named but is necessary
Technical Specs:

- Xen virtual server
- 1 vCPU
- 384 MB memory
- 5GB vDISK

Responsible for:

- All e-mail traffic which are generated by the machines for maintenance/error reports and email generated by the Administration Interface

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- MailScanner
- Spammassassin
- Procmail
- Baruwa webfrontend
- Apache2
- MySql

Management,Backup,Monitoring Server

ServerName: filippoturati<following number>.objectrepository.eu
Role(s) High Level Design: not named but is necessary for server monitoring
Technical Specs:

- Xen virtual server
- vCPU
- 1024 MB memory
- 5GB vDISK + 50GB vDISK

Responsible for:

- Delivering a monitoring platform (logging, error reporting, other IT-related reporting), patch management

Used software:

- Ubuntu LTS 10.04.x 64bit
- Webmin administrator interface
- Shell In A Box
- Sendmail
- Secure Shell Deamon
- ZenOss (management tool)
- Bacula (Backup)
- Puppet (Linux patch management)

3.5.2 Converter Environment

Citrix XenServer

ServerName: antonpannekoek<following number>.objectrepository.eu
Role(s) High Level Design: Not named but is necessary
Technical Specs:

- Server
- QC CPU
- 8GB memory
- 72GB DISK RAID 1

Responsible for:

- Hosting all virtual Converter servers

Used software:

- Citrix XenServer 5.6.xxx

Conclusion

In this document we have described the high and low level design of the SOR. The input for this design came from:

- The High Level Design WP2 (T2.1)
- Gathered requirements from the Content Providers (CP) in the "HOPE consortium"
- Milestone document M5.1 – Repository workflow and Requirements specification

All components of the SOR have been specified including the low level design is specified.
This document will evolve during the following releases of the SOR and PID Service. Some of the technology choices can change. When the requirements, technology choices or specifications change, this document will be updated.

Appendix A - Example HOPE Persistent Identifier Web service interface

See document: Deliverable D5.1 Supplement – Repository Infrastructure and Detailed Design Appendixes

Appendix B – Low Level Design

See document: Deliverable D5.1 Supplement – Repository Infrastructure and Detailed Design Appendixes

Appendix C – Organizations providing parts of the infrastructure of the SOR

See document: Deliverable D5.1 Supplement – Repository Infrastructure and Detailed Design Appendixes

Appendix D – Technical Glossary SOR

See document: Deliverable D5.1 Supplement – Repository Infrastructure and Detailed Design Appendixes