



Project no. 033572

CASPAR

Cultural, Artistic and Scientific knowledge for **P**reservation, **A**ccess and **R**etrieval

Instrument: Information Society Technologies

Thematic Priority: 2.5.10 Access to and preservation of cultural and scientific resources

CASPAR Integration with External Systems



Document identifier:	CASPAR-3201-RP-0101-1_0
Submission Date:	21-10-2009
Due date:	30-09-2009
Work package:	3200
Partners:	All Partners
WP Lead Partner:	ACS
Document status	FINAL

Abstract: This document provides a brief report of integration with external systems.



Delivery Type Report
 Author(s) CASPAR Consortium

 Approval David Giaretta
 Summary
 Keyword List
 Availability PUBLIC

Document Status Sheet

Issue	Date	Comment	Author
0_1	01 Sept 2009	Initial draft bringing together contributions	Fulvio Marelli
1_0	21 Oct 2009	Add material collected form testbeds	David Giaretta





Project information

Project acronym:	CASPAR
Project full title:	Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval
Proposal/Contract no.:	IST-2006-033572

Project Officer: Martin Mühleck

Address:	<p>INFSO-E3 Information Society and Media Directorate General Content - Learning and Cultural Heritage</p> <p>Postal mail: Bâtiment Jean Monnet (EUFO 1167) Rue Alcide De Gasperi / L-2920 Luxembourg</p> <p>Office address: EUROFORUM Building - EUFO 1167 10, rue Robert Stumper / L-2557 Gasperich / Luxembourg</p>
Phone:	+352 4301 31531
Fax:	+ 352 4301 33190
Mobile:	
E-mail:	Martin.MUEHLECK@ec.europa.eu

Project Co-ordinator: David Giaretta

Address:	STFC (formerly STFC), Rutherford Appleton Laboratory Chilton, Didcot, Oxon OX11 0QX, UK
Phone:	+44 1235 446235
Fax:	+44 1235 446362
Mobile:	+44 (0) 7770326304
E-mail:	david.giaretta@stfc.ac.uk





Content

1	INTRODUCTION.....	6
1.1	Threats to digitally encoded information	7
1.2	Preservation Workflow	8
1.3	CASPAR answers to the threats.....	14
2	DEPLOYMENT SCENARIOS.....	16
3	THE PDS INTEGRATION	21
3.1	Glossary	21
3.2	PDS overview	22
3.3	PDS architecture.....	23
3.4	PDS interfaces	25
3.5	Integrating PDS with existing archives.....	26
3.6	Examples.....	32
3.7	StorageHandler interface	35
3.8	PDS conclusions.....	38
3.9	PDS references	39
4	THE ESA DEVELOPMENT.....	40
4.1	Possibility of applying the CASPAR features to ESA (and other) data.....	40
4.2	The GENESI-DR project.....	40
4.3	The CASPAR and GENESI-DR combined approach.....	42
4.4	Studies concerning the integration between CASPAR & GRID	45
4.5	CASPAR and GRIDIFICATION.....	49
4.6	GRID GLOSSARY	52
5	INTEGRATION REPORT FROM IRCAM.....	53
5.1	CASPAR customisation by IRCAM.....	53
5.2	Overview of Integrated CASPAR Functionalities.....	54





6	UNESCO INTEGRATION.....	57
6.1	EWE: an extensible tool for the preservation of VHRP data.....	57
7	CONCLUSIONS	60





1 INTRODUCTION

CASPAR can be decomposed in a set of components providing OAIS compliant functionalities for long term data preservation. Those components can work independently or in an integrated framework with other CASPAR components or third party data preservation systems. Some of the CASPAR project activities were focused on the integration of some of those functionalities into pre-existing preservation services and/or facilities. This document contains the description of the analysis and the development performed by the partners to achieve the CASPAR components integration with external preservation system. Figure 1 provides an overall view of the examples provided here.

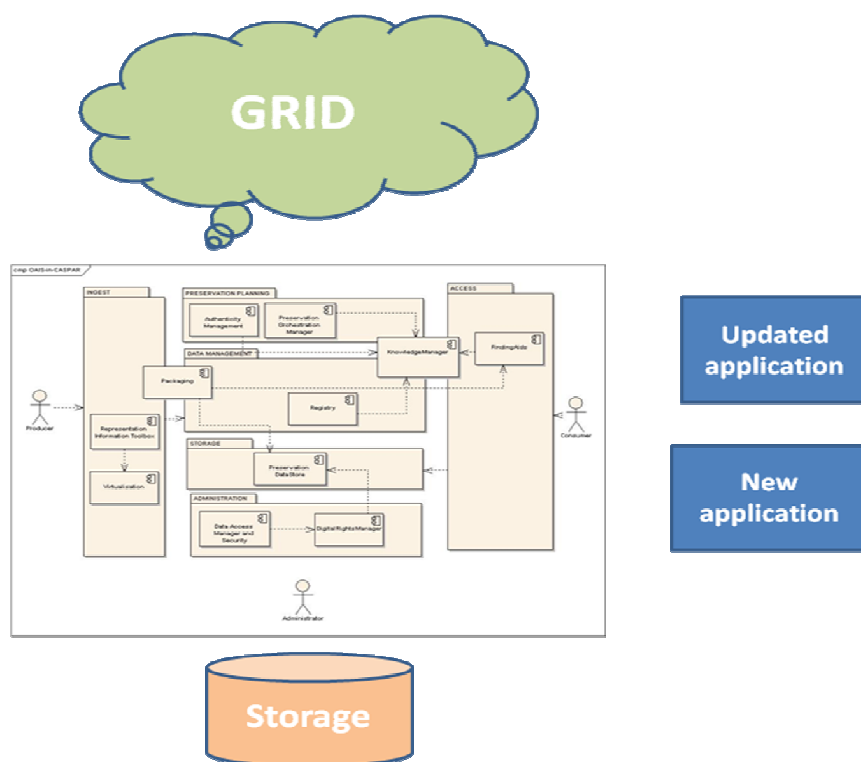


Figure 1 Overall view

In chapter two there is a brief recap of some of the ways in which CASPAR can be integrated into existing repositories, ranging from strengthening the OAIS functional model components of a repository to a very distributed set of repositories and services which together preserve digital information.

The third chapter refers to the work done in IBM – Haifa to integrate the Preservation Data Store into a generic external preservation system and also integration with the iRODS system.

Following this we describe the studies, development and future integration plans that have been performed by the ESA-ACS staff. This describes the integration of CASPAR components into the GENESI-DR project, and looks at some more general GRID-related aspects.

The next chapter refers to the integration work performed by IRCAM to integrate CASPAR into the MUSTICA server, obtaining the MUSTICASPARE platform. This is an example where an existing well used application (Mustica) has been modified to include CASPAR capabilities – thus producing MustiCASPAR. The CASPAR capabilities are provided subtly in the updated





interface so that they do not interfere with normal users. Following this we describe some of the work within UNESCO in producing an application specific to their needs.

In the next sub-section we review a number of the other ways in which the CASPAR components can form part of the general infrastructure which can help to support digital preservation which is the most general type of distributed deployment scenario.

1.1 THREATS TO DIGITALLY ENCODED INFORMATION

In general terms one can list the changes which threaten digitally encoded information are:

Hardware changes
Software changes
Environment changes, including for example the legal environment
Knowledgebase of the Designated Community changes

Surveys have been undertaken by PARSE.Insight and members of the Alliance for Permanent Access, investigating creation, re-use, preservation and publication of digital data. These surveys show a substantial demand for a science data infrastructure which is consistent across nations, continents and over a remarkably wide range of disciplines.

There has been time for only an initial analysis of the results. The results of most immediate interest revolve around a collection of “threats” to digital preservation which are based on prior analyses of the domain and which are pertinent to data re-use also. It is worth noting that similar lists can be found in most project proposals related to digital preservation.

The major threats are as follows:

Users may be unable to understand or use the data e.g. the semantics, format, processes or algorithms involved
Non-maintainability of essential hardware, software or support environment may make the information inaccessible
The chain of evidence may be lost and there may be lack of certainty of provenance or authenticity
Access and use restrictions may not be respected in the future
Loss of ability to identify the location of data
The current custodian of the data, whether an organization or project, may cease to exist at some point in the future
The ones we trust to look after the digital holdings may let us down

The preliminary survey results show that between 50% and 70% of responses indicate that all the threats are recognized as either “Important” or “Very Important”, with a majority supporting the need for an international preservation infrastructure.

From these threats one can propose a number of general requirements for solutions to them.

Threat	Requirements for solutions
Users may be unable to understand or use the data e.g. the semantics, format, processes or algorithms involved	Ability to create and maintain adequate Representation Information





Non-maintainability of essential hardware, software or support environment may make the information inaccessible	Ability to share information about the availability of hardware and software and their replacements/substitutes
The chain of evidence may be lost and there may be lack of certainty of provenance or authenticity	Ability to bring together evidence from diverse sources about the Authenticity of a digital object
Access and use restrictions may make it difficult to reuse data, or alternatively may not be respected in future	Ability to deal with Digital Rights correctly in a changing and evolving environment
Loss of ability to identify the location of data	An ID resolver which is really persistent
The current custodian of the data, whether an organisation or project, may cease to exist at some point in the future	Brokering of organisations to hold data and the ability to package together the information needed to transfer information between organisations ready for long term preservation
The ones we trust to look after the digital holdings may let us down	Certification process so that one can have confidence about whom to trust to preserve data holdings over the long term

1.2 PRESERVATION WORKFLOW

Figure 2 shows one of the views that CASPAR takes of the process needed when information is ingested into a repository and then taken out at some time in the future.



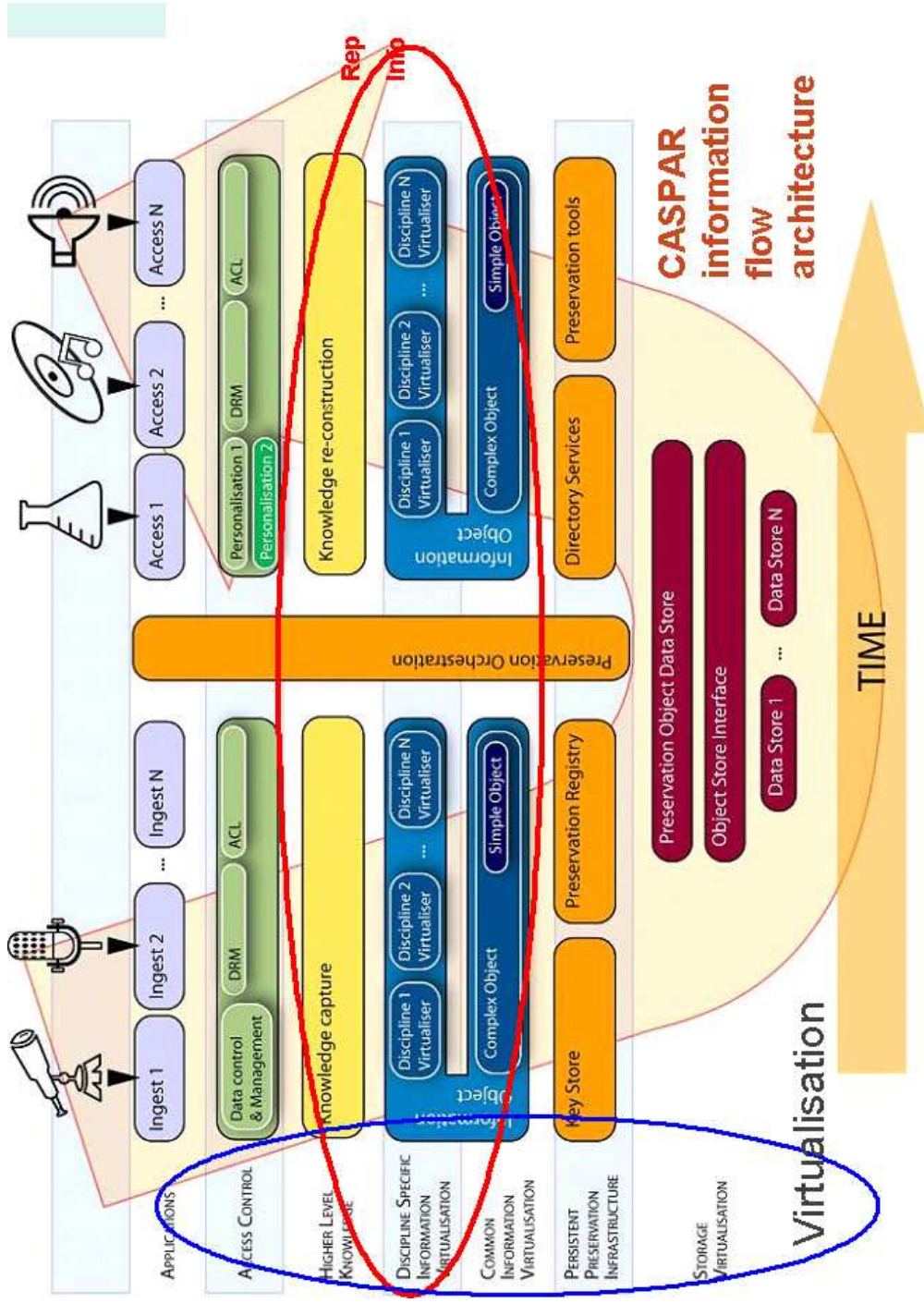


Figure 2 CASPAR Information Flow





Many details must be captured including
access rights, Digital Rights Management (DRM) and Access Control Lists (ACL)
Representation Information of various types
high level knowledge

various types of descriptions including a the way in which complex objects may be viewed as a composite of simpler objects. Some of these objects may be discipline specific whereas others are rather general.

For example an image is a fairly general concept – essentially an array of numbers, whereas an Astronomical image is an image plus an astronomical co-ordinate system and a way to map to physical measurements.

Details of the simple objects down to the bit level must also be captured.

Note that here, as well as elsewhere, virtualisation techniques can be applied. Further details of this and many other aspects of preservation can be found on the CASPAR web site and in particular the CASPAR Conceptual Model (CASPAR, 2007).

The digital objects must be stored, indicated here as a Preservation Object Data Store.

Subsequently the process must be reversed, for example:

Information must be extracted using the Representation Information at various levels

Access constraints must be understood and respected

It is worth noting that much of these descriptions and extra pieces of information (metadata) will themselves be digitally encoded and will therefore also need to be preserved, using the same techniques.

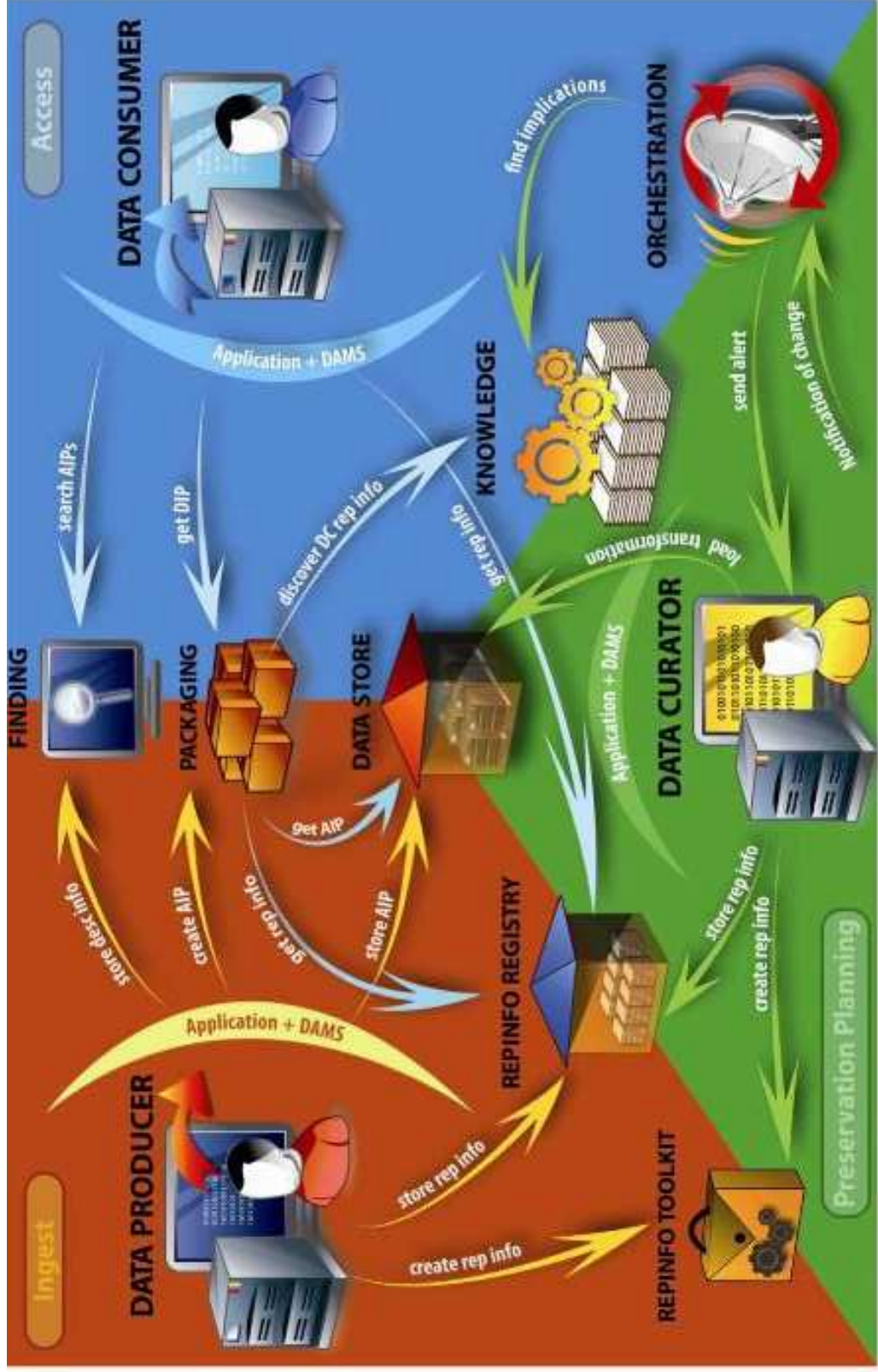
As the final part of this brief overview of the CASPAR approach, some major workflows involved in preservation are described here. As described in the CASPAR Conceptual Model [D1201], preservation is a complex task and almost certainly requires, over time, the combination of efforts from many participants.

A number of other workflows arise from the support components identified by CASPAR and the UK Digital Curation Centre (<http://www.dcc.ac.uk>), which may be summarized in Figure 3





Figure 3
Presentation
Workflow





The Key Components shown in Figure 3 are described next. Additional details may be found in [D1201] and [D1301].

1.2.1 Discipline independent components

1.2.1.1 Registries of Representation Information (RepInfo Registry in the diagram)

The prime functions of a Registry/Repository are:

Given an identifier of a piece of Representation Information (RepInfo), return that piece of Representation Information to the requestor. This Representation Information will in general be an opaque binary object as far as the Registry/Repository is concerned.

Allow searching of the holdings of the Registry in order to enable the re-use of existing RepInfo.

To facilitate this searching, each piece of RepInfo should be classified under one or more Classification Schemes, and have a searchable text description of the RepInfo.

Each piece of RepInfo should itself have a pointer to its own RepInfo, and also details of its PDI.

The Registry/Repository should itself be an OAIS which can be certified for long-term preservation of information.

1.2.1.2 Orchestration manager (Orchestration in the diagram)

The Orchestration component has to:

allow individuals to register their interests and expertise

collect information from (anonymous or registered) individuals about changes in software, hardware, environment or Knowledge Base of any Designated Community. This information will be passed on to the RepInfo Gap Manager component.

receive information from the RepInfo Gap Manager component about a gap which has been identified

send requests to appropriate registered users, based on their interests and expertise, for the creation of required Representation Information

1.2.1.3 RepInfo Gap Manager (Knowledge in the diagram)

The RepInfo Gap Manager component embodies a small but essential application of Knowledge Management techniques to preservation. Its main purpose is to assist in identifying gaps which have arisen as a result of changes in hardware, software, environment and Knowledge Bases of Designated Communities.

The changes are notified by human participants in the preservation process. The RepInfo Gap Manager knows of the existing dependencies between pieces of Representation Information, working closely with one or more instances of the Registry/Repository. The labels in the Registry/Repository capture those dependencies. The changes imply that gaps in the Representation Information network will have arisen, which must be filled. Human participants must be alerted and requested to provide new Representation Information to fill those gaps. The human participation may not always be necessary; the RepInfo Gap Manager may be able to bring in Representation Information from another, existing, source to fill the gap – although this would have to be checked by humans.

1.2.1.4 Digital Object Storage

The Digital Object Storage (or sometimes simply “Storage”) component takes care of the “Digital Object” and encapsulates:





The secure preservation of the bits which encode the information of interest. This of course applies to a primary Data Object, Representation Information, Preservation Description Information etc, the latter also being Data Objects. These **individual** stored objects form the simplest element in the storage system, and each needs only be regarded as opaque binary objects, whose internal structure need not be known or understood by the Storage system, although the structure of the AIP, e.g. how to get the PDI object out of the AIP, will be known to it.

The association of Representation Information and PDI with the Content Information. This association may include having copies of the Representation Information or PDI kept within the Storage system. However it is important to recognise that neither of these can be complete. For example the Representation Information Network will change as, for example, the Knowledge Base of the Designated Community changes. Similarly the Provenance information will include not just the technical information about copying but also but also include descriptions of various real-world entities (eg. persons, organisations and their attributes, roles and actions) whose social context is also associated with the data. Therefore both Representation Information and PDI will have to include a pointer out of the storage system.

The automatic maintenance of the *technical* provenance information, including details of what are essentially internal events including copying, replication and refreshment and the objects.

The policies which the archive imposes on the stored objects (and the Representation Information, PDI etc associated with the encoded instances of these policies), for example

the number of backup copies, offsite and on-site, on-line and near-line, and replication

the access controls

the distribution of information among the individual pieces of virtualised storage

maintenance of namespaces

maintenance of collection level information

The ability to hand on the stored AIPs, and appropriate collection information, to another OAIS system – either because of technological change or because of organisational change as the preserved information is passed on to the next in the chain of preservation.

1.2.1.5 Packaging

Creates Information packages, in particular Archival Information Packages.

1.2.1.6 Finding Aid (Finding in the diagram)

Additional Finding aids which allow AIPs to be searched for in a repository.

1.2.2 Toolkits

1.2.2.1 ReplInfo Toolkit

This toolkit is essentially an open-ended collection of individual tools which may be used to create Representation Information.

1.2.2.2 Authenticity Toolkit (not shown in the diagram)

Another collection of tools which allow the capture of evidence which may be used to judge authenticity

1.2.2.3 DRM and Access Rights Tools (not shown in the diagram)

Tools to virtualise Digital Rights and Access Rights so that they can be preserved independently of their particular encoding.





Figure 3 contains a number of information flows; some sequences of these flows making up workflows important for digital preservation and two of these are described next.

1.3 CASPAR ANSWERS TO THE THREATS

The following table maps the CASPAR components and toolkits to the solutions to the threats.



**Table 1 CASPAR solutions to threats**

Threat	Requirements for solutions	CASPAR Component
Users may be unable to understand or use the data e.g. the semantics, format, processes or algorithms involved	Ability to create and maintain adequate Representation Information	RepInfo toolkit, Packager and Registry – to create and store Representation Information. In addition the Orchestration Manager and Knowledge Gap Manager help to ensure that the RepInfo is adequate.
Non-maintainability of essential hardware, software or support environment may make the information inaccessible	Ability to share information about the availability of hardware and software and their replacements/substitutes	Registry and Orchestration Manager to exchange information about the obsolescence of hardware and software, amongst other changes. The Representation Information will include such things as software source code and emulators.
The chain of evidence may be lost and there may be lack of certainty of provenance or authenticity	Ability to bring together evidence from diverse sources about the Authenticity of a digital object	Authenticity toolkit will allow one to capture evidence from many sources which may be used to judge Authenticity.
Access and use restrictions may make it difficult to reuse data, or alternatively may not be respected in future	Ability to deal with Digital Rights correctly in a changing and evolving environment	Digital Rights and Access Rights tools allow one to virtualise and preserve the DRM and Access Rights information which exist at the time the Content Information is submitted for preservation.
Loss of ability to identify the location of data	An ID resolver which is really persistent	Persistent Identifier system: such a system will allow objects to be located over time.
The current custodian of the data, whether an organisation or project, may cease to exist at some point in the future	Brokering of organisations to hold data and the ability to package together the information needed to transfer information between organisations ready for long term preservation	Orchestration Manager will, amongst other things, allow the exchange of information about datasets which need to be passed from one curator to another.
The ones we trust to look after the digital holdings may let us down	Certification process so that one can have confidence about whom to trust to preserve data holdings over the long term	The Audit and Certification standard to which CASPAR has contributed will allow a certification process to be set up.





2 DEPLOYMENT SCENARIOS

A number of strategies were described near the beginning of the projects and it is useful to mention them again here.

We argue in D4104 that CASPAR components help repositories support the OAIS Information Model.

In addition show below the mapping of OAIS components to the OAIS Functional Model – this indicated where components can sit within a repository to aid its information preservation aims.

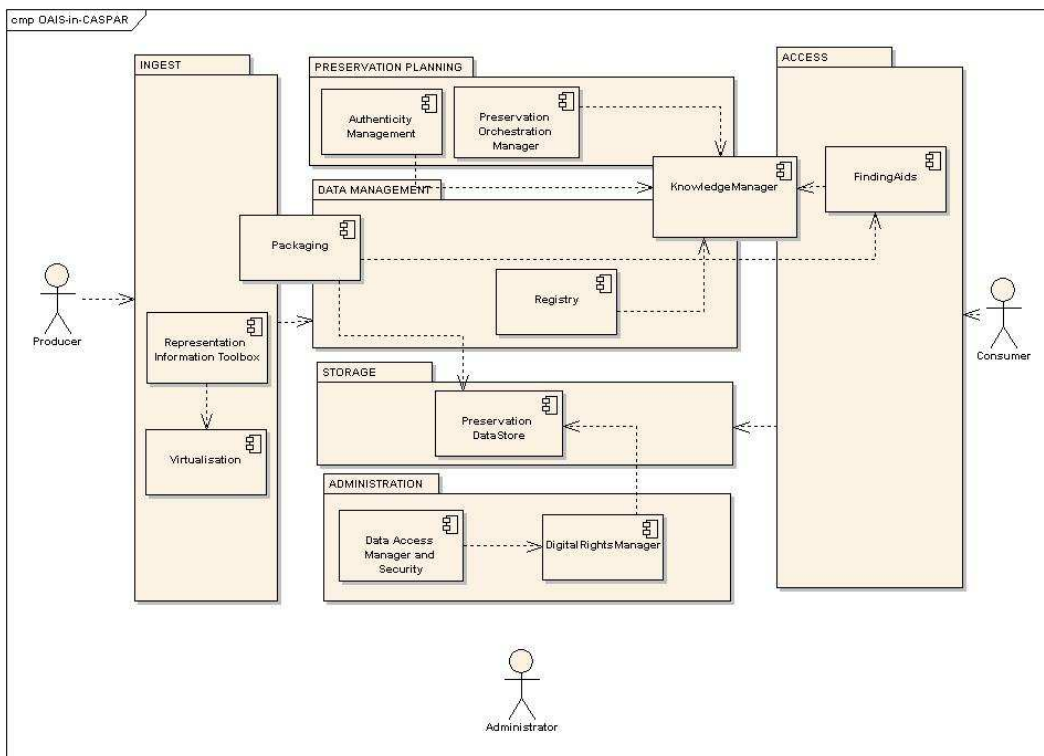


Figure 4 Mapping CASPAR components to OAIS Functional Model

Another, slightly more detailed view is shown below.



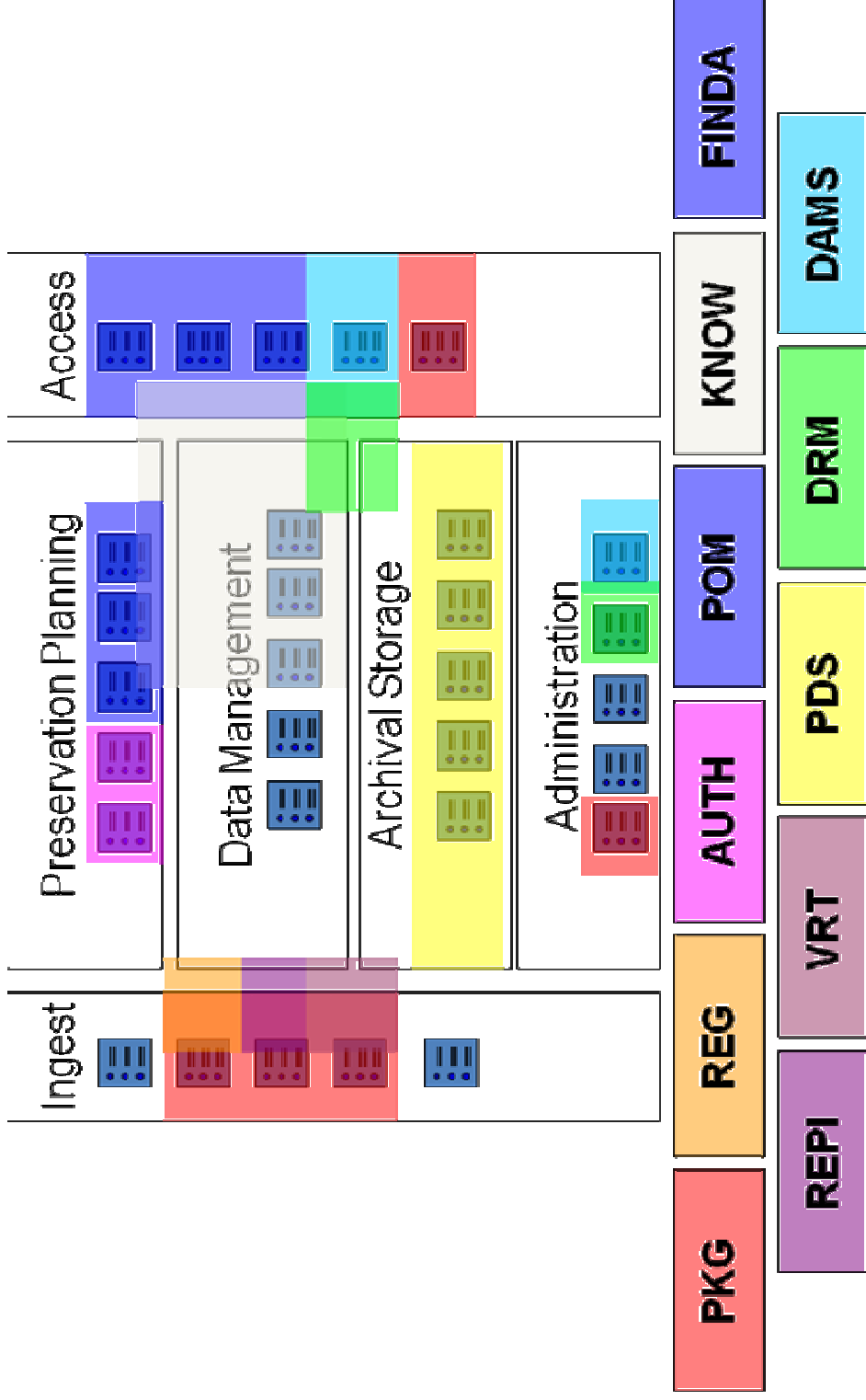


Figure 5 More detailed mapping of CASPAR components to OAIS Functional Model

In addition we can show some of the possible deployments of OAIS components in single and multiple repositories.



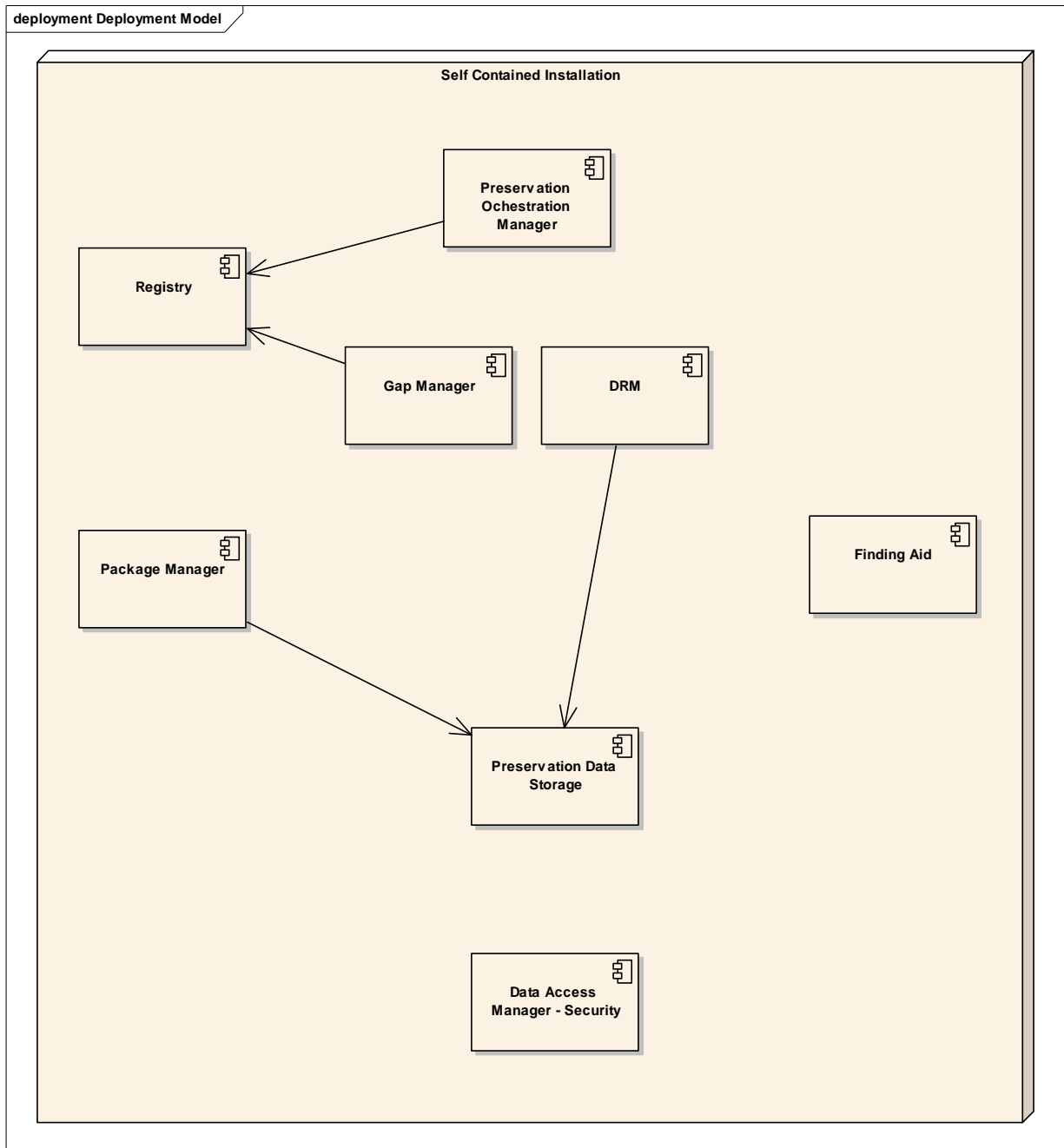


Figure 6 Self contained scenario

This is a self-contained situation where a repository wished to bear the full responsibility for preservation, perhaps because of confidentiality considerations.

The next two models show situations where there is a shared service. Figure 7 shows the minimal shared service – namely sharing the Registry/Repository of Representation Information.

Figure 8 shows a fuller sharing of services between several repositories.



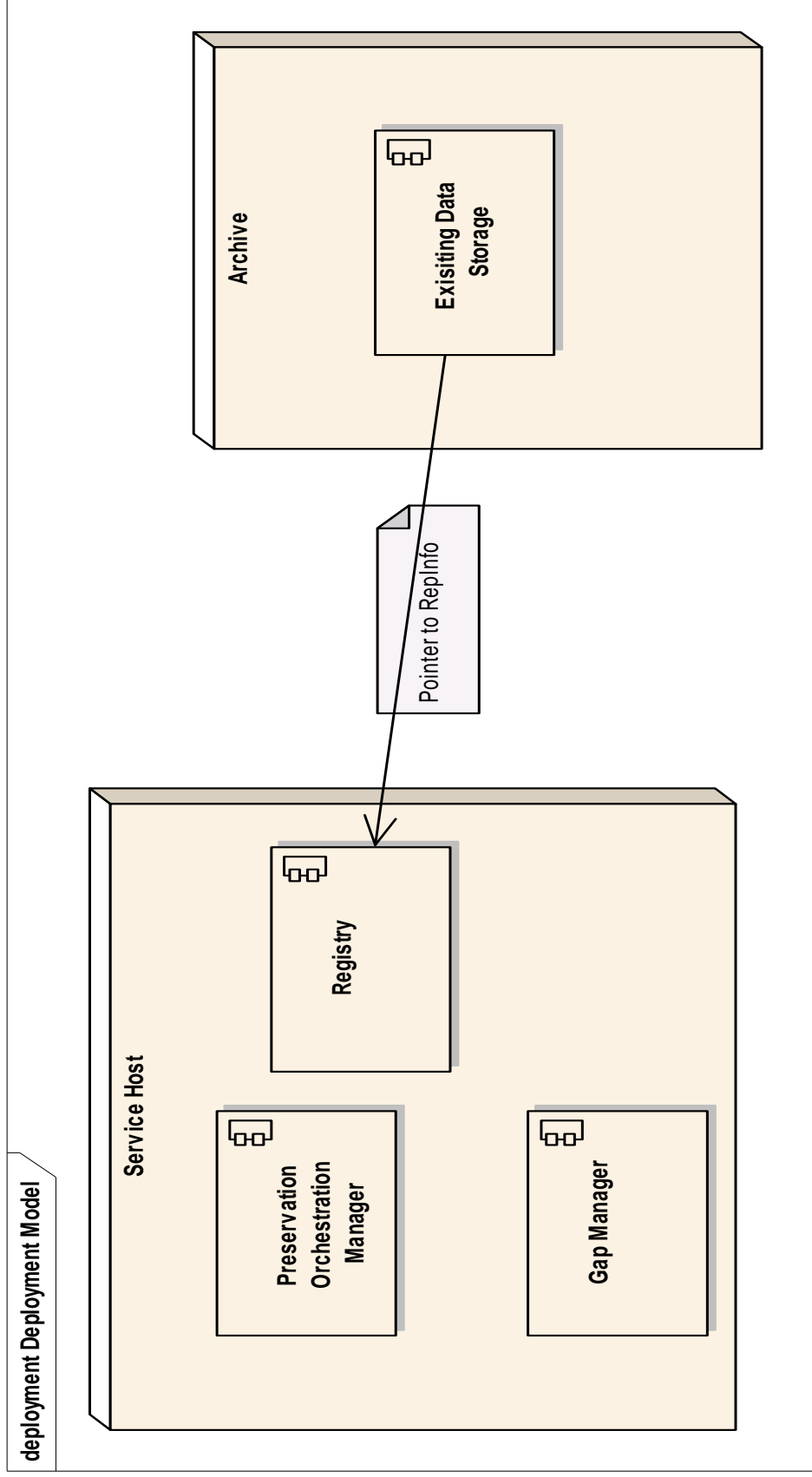


Figure 7 Minimal shared service installation



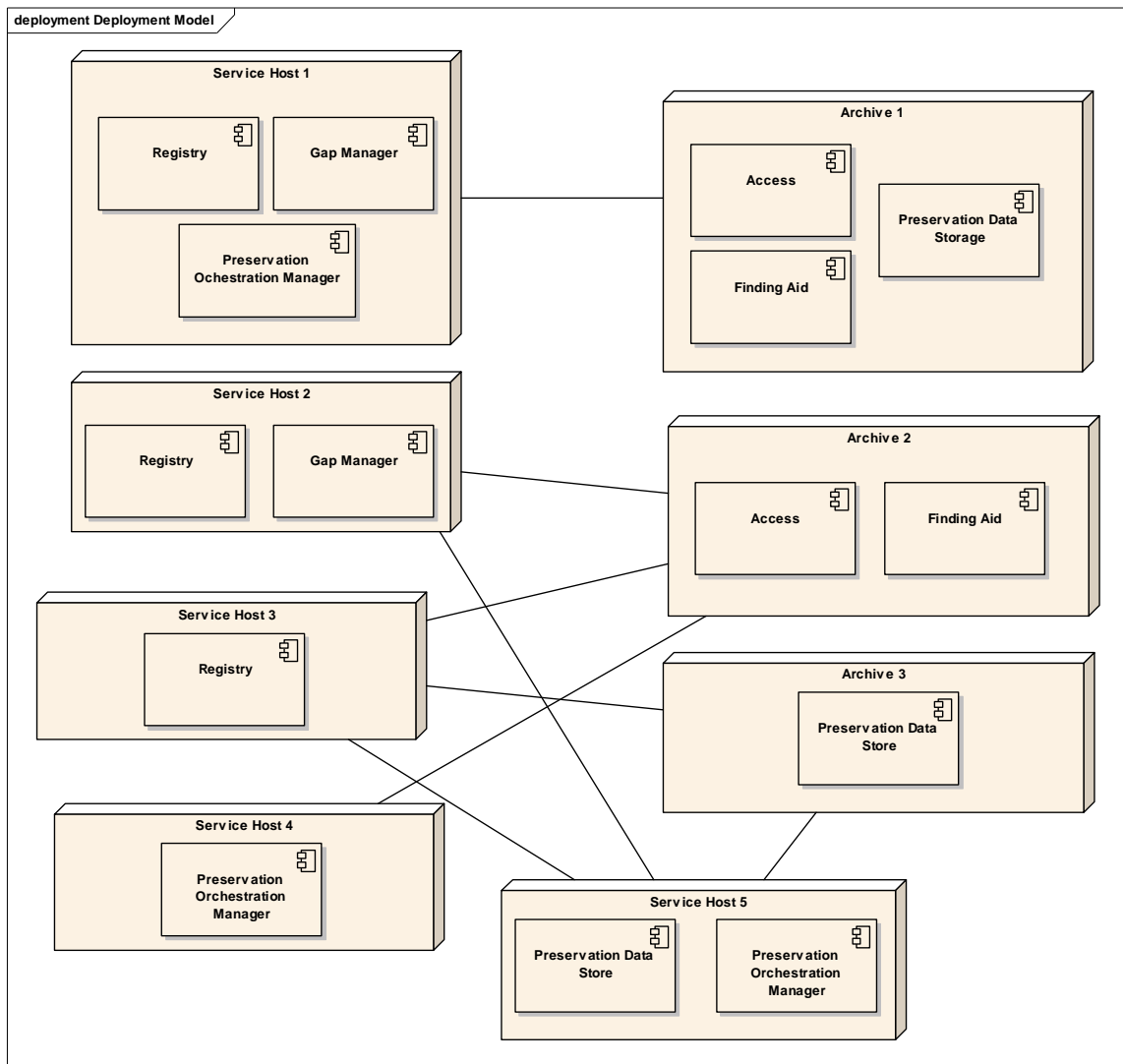


Figure 8 Shared service installation





3 THE PDS INTEGRATION

The purpose of this section is to propose a design for integrating Preservation DataStores (PDS), the archival storage component in CASPAR, with an existing system.

Such integration may vary from one existing system to another.

The main goal we wish to achieve through this document is progress towards integrating PDS into CASPAR data holders' existing storage systems.

This document relies on the PDS public deliverable for CASPAR, D2201 - Preservation DataStores interface, which can be found at

<http://www.casparpreserves.eu/publications/deliverables>.

First we bring an overview of PDS, its architecture and interfaces (Section - PDS overview).

Then, the main section of this section describes why and how can PDS integrate with an existing system; the basic integration architecture and the possible variants, and details about the work that needs to be done to implement such integration (see Integrating PDS with existing archives)

Following are examples for existing systems that PDS may integrate with (section 4), and the document ends with conclusions (section 5).

3.1 GLOSSARY

Bit preservation	The processes used to ensure that the bits comprising a preserved object are not lost or corrupted over time. These processes include refreshing, backups, and error correcting code modules.
CASPAR	Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval.
Content data object	The 'raw' data that makes up the content to be preserved.
Content information object	The raw data and the metadata needed to interpret it; namely, the content data plus its representation information.
CDO	The Content Data Object that contains the raw data.
Context	The relationship of the content information to its environment, initially as perceived by the content data object provider. Later on, more context information may be implied or derived from the preservation process.
Designated community	The primary OAIS user group that needs to access and understand the information preserved in the OAIS system. This means that the OAIS must have an appreciation of the community's knowledge base.
Digital preservation	The series of managed activities necessary to ensure continued access to digital materials for as long as necessary.
DROID (Digital Record Object Identification)	A software tool developed by The National Archives (UK) to perform automated batch identification of file formats. DROID uses internal and external signatures to identify and report the specific file format versions of digital files. These signatures are stored in an XML signature file, generated from information recorded in the PRONOM technical registry (see PRONOM). http://droid.sourceforge.net
Fixity	The information that documents the authentication mechanisms and provides authentication keys to ensure that the content information object has not been altered in an undocumented manner.





Logical preservation	The processes used to ensure the understandability and usability of the data, in spite of the unknown changes in technologies and users in the future.
Metadata	Information about the content data object that is needed for its preservation.
Migration	The act of moving data from one system to another because of a change.
OAIS	Open Archival Information System. An ISO standard (ISO 14721:2003 OAIS) that specifies a reference model for an archive, consisting of an organization of people and systems that have accepted the responsibility to preserve information for a designated community.
Preservation system	An archiving system in which the lifetime of the data it needs to store exceeds the lifetime of the program/format with which the data is interpreted and the lifetime of the media that stores the bits.
Preservation aware storage	The storage component of a digital preservation system that has built-in support for preservation.
Preservation DataStores (PDS)	A new OAIS-based preservation aware storage. It will also serve as the storage component of CASPAR infrastructure.
PRONOM	An on-line information system about data file formats and their supporting software products. Originally developed to support the accession and long-term preservation of electronic records held by the National Archives, PRONOM is now being made available as a resource for anyone requiring access to this type of information. http://www.nationalarchives.gov.uk/pronom
Provenance	The information that documents the history of the content information. This information describes the origin or source of the content information, any changes that may have taken place since it was originated, and who has had custody of it since it was originated.
Representation information	The information that is required to interpret the content data object (raw data) into more meaningful concepts (ultimately more meaningful for humans).

3.2 PDS OVERVIEW

Preservation DataStores (PDS) are OAIS-based preservation aware storage [1, 2, 3] that focus on supporting logical preservation. They are aware of the structure of an archival information package (AIP) and offload OAIS derived generic functions such as representation information inclusion, provenance tracking, fixity computation, and migration support to the storage layer.

They provide strong encapsulation of large quantities of metadata with the data at the storage level and enable easy migration of the preserved data across storage devices.

PDS aims to address the following preservation-aware storage requirements:

In the storage, encapsulate and physically co-locate the raw data and its complex interrelated metadata objects, such as representation information, provenance, and fixity. This ensures that the metadata needed for interpretation is not separated from the raw data and thus never lost (assuming the raw data survives).

Include the representation information of metadata (e.g., representation information of fixity and provenance) so the metadata can be interpreted when accessed in the future.

Utilize the locality property and execute data intensive functions such as fixity (i.e., data integrity) computations within the storage component. This will lessen the network bandwidth and reduce the risks of data loss.





Handle some of the provenance events internally. The applications on top of the preservation aware storage should be free of managing events that can be handled internally in the storage. Moreover, this enables richer types of provenance events and the inclusion of events related to the migration between physical medium and the transformation of representations.

Support the loading and execution of external transformations during the migration process and facilitate on demand triggering of these transformations.

Support media migration, as opposed to system migration. In media migration, performing migration from one system to another can be done by physically detaching the media from one system and attaching it to the new system.

Maintain referential integrity including updating all the links during the migration process so they remain valid in the new system. This requires an awareness of certain metadata fields that represent links, both internally to the system and externally.

Ensure readability of the data by a different system in the future. This is done by developing and supporting global self-describing media independent formats.

Support a graceful loss of data. Some portions of the data are likely to be lost or become corrupted over time. If some data is lost, a good preservation system must minimize the economic effect of this data loss and prevent cases where data that is still intact in the system cannot be read or interpreted.

3.3 PDS ARCHITECTURE

The PDS architecture [4] includes a stack of three layers based on OAIS, XAM, and OSD. Figure 9: *Preservation DataStore architecture* depicts the general architecture of PDS. The top layer is an OAIS-based preservation engine, which provides preservation functionality for heterogeneous data and applications. It includes efficient generation and placement of AIPs along with support for migration and data transformations performed within the storage.

The second layer includes an eXtensible Access Method (XAM) [5] library. XAM is an emerging storage standard intended for reference information that provides a logical abstraction for data containers. The bottom layer includes an Object-based Storage Device (OSD) [6, 7, 8], an advanced type of storage implementing an object-based interface that has a built-in access control mechanism.

In OAIS, AIPs are the information objects that are passed to and from the storage component. Thus, in PDS, AIPs are the main objects in the interface. While PDS has a generic interface, our current implementation supports both a direct API as well as a web services API. We chose a standards-based web services API because web services are platform independent and support clients built within different environments; these are important features in preservation environments.



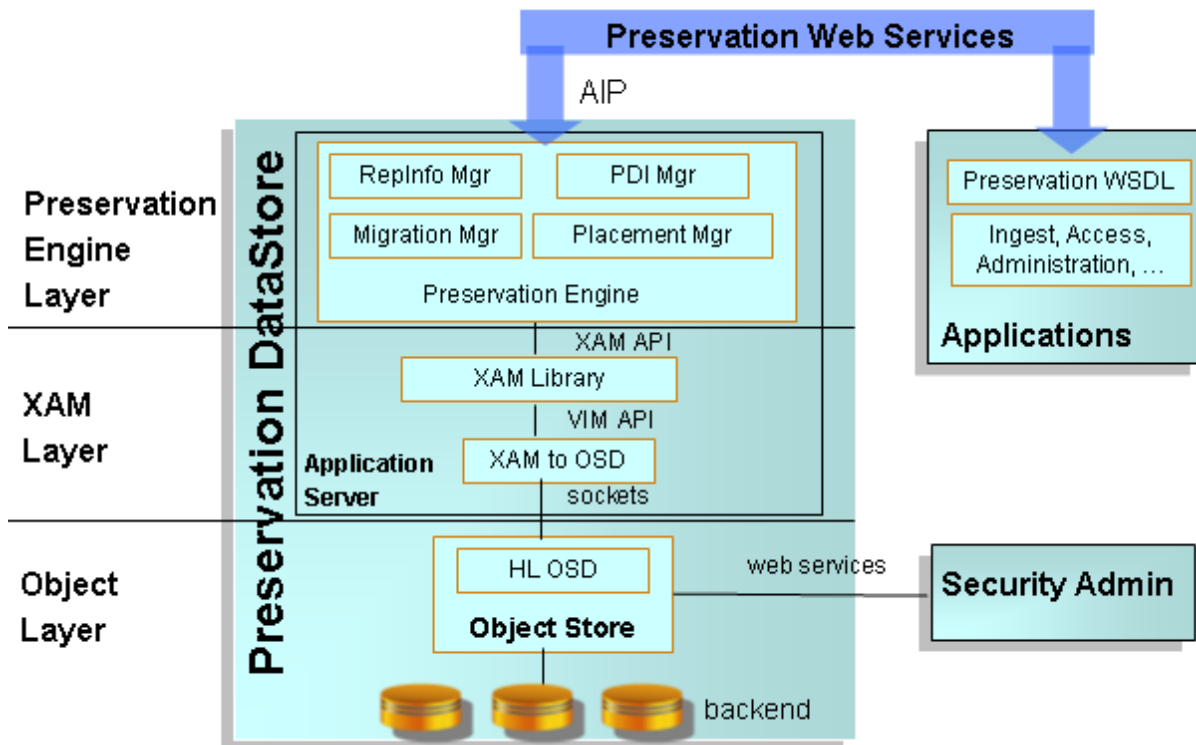


Figure 9: Preservation DataStore architecture

3.3.1 Preservation Engine layer

The preservation engine component provides the external API, implements the OAIS abstractions and provides the preservation characteristics.

The unique features that the preservation engine implements include:

Managing availability/data loss – being aware of the AIP structure, and according to specific policies, the preservation engine may group data objects and create copies both within a medium and across media.

AIP transformations – in addition to media migrations there may be a need for data transformations that PDS supports to be carried out inside the storage thus allowing optimal scheduling and minimal data transfers. This may be the only way to make the execution of massive data transformations feasible. Transformations may also be performed on metadata sections.

AIP identifier generation – satisfies cases in which an AIP is ingested without a unique identifier and also cases in which PDS generates AIPs internally (e.g., preserving the result of data transformation, or preserving RepInfo that was embedded in an ingested AIP).

Storelet container - a module container that can embed and execute restricted modules with pre-defined interfaces, used to perform data intensive functions in the storage (e.g., data validation, data transformation, or authenticity processes).

Manage preservation specific metadata - fixity computations, documenting provenance events, managing RepInfo, etc.

In addition to these specific features, a major role of the preservation engine layer is to perform mapping between the different layers of abstraction. At the top, the preservation engine layer uses OAIS concepts to communicate with its clients. In turn, the preservation engine builds





upon the XAM underneath to implement its function. Thus it must map between the OAIS and XAM levels of abstraction.

3.3.1.1 XAM layer

While PDS exposes high level complex logical objects such as AIP, existing storage systems (object, file or block) expose much lower level interfaces and do not provide the necessary abstractions. Our architecture uses a middle layer of abstraction to mediate between the lower storage system and the preservation engine. We chose to base this mid-layer on Extensible Access Method (XAM).

XAM is a Storage Networking Industry Association (SNIA) initiative to define a standard interface between consumers (application and management software) and providers (storage systems). The XAM specification defines three primary objects: XSet, XSystem and XAM Library. The XSet, which is the fundamental artifact in XAM, is the basic unit of data for application to commit to persistent storage. It is a data structure that is a package of multiple pieces of data and metadata, bundled together for access under a common globally unique external name, called an XUID. The XSystem, a logical container for one or more XSet records, serves as virtual storage for XAM applications. The XAM Library enables applications to discover and communicate with XSystems.

3.3.1.2 Object layer

The XAM standard provides us with a high level storage abstraction needed for the preservation engine. The XAM library interacts with the supplied underlying storage system via the Vendor Interface Module (VIM) API. The VIM maps XAM entities such as XSystems, XSets, and XSet fields and their attributes to the vendor storage system entities. Currently we support Object Store (OSD) as the object layer implementation of PDS. XAM communicates with it via XAMtoOSD component, an implementation of the VIM API that maps XAM to OSD.

3.4 PDS INTERFACES

The PDS interfaces expose the PDS entry points and a set of interfaces to support these entry points. The entry points may be called directly or via web services. The PDS interfaces aim to be abstract, to be independent of technology, and to survive implementation replacements.

The PDS entry points may be divided into groups; each group of entry points is responsible for a different PDS function and may include several variants on that function:

- ingest AIP - implements different ways to ingest an AIP or an AIC to PDS

access AIP - implements access either to a complete AIP or to different sections of an AIP

handle AIP - enables manipulating AIPs that were previously ingested

query AIP - performs queries on the AIPs in the system

load storlet - loads modules into PDS to be executed later

migrate AIP - performs migrations and transformations on AIPs

handle policies - enables manipulating PDS policies

informative entry points - provide information about the PDS system





3.5 INTEGRATING PDS WITH EXISTING ARCHIVES

This section describes the integration of Preservation DataStores (PDS) with existing archive and suggests how to support data that already resides in existing file systems and archives but needs to be preserved for decades and longer.

This integration may have different variants, depending on the existing system; each offers different parts of PDS to be deployed, and may include additional components to bind PDS to the system.

3.5.1 Motivation

In many cases, the data that is subject to long term digital preservation already resides in existing archives. These archives may be simple file systems or more advanced archives that include enhanced functions such as metadata advanced query, hierarchical storage management, routine or special error checking, disaster recovery capabilities, and bit preservation. Some of this data is generated by applications that are unaware of the OAIS specification and the AIP logical structure, and generally include just the raw content data with minimal metadata. While these archives are appropriate for short term data retention, they cannot ensure long term data interpretation at some arbitrary point in the future; at this point, everything can become obsolete including hardware, software, processes, format, people, and so forth.

PDS was designed as a stand-alone archival storage component of an OAIS preservation system. However, when adding preservation capabilities to an archive by deploying PDS, it is unlikely that the existing storage system will be replaced with PDS. Instead, there is a need to enable integrating PDS into the existing system; thereby adding its long term preservation capabilities to the existing storage capabilities.

Today, more and more storage systems have a strong need to add preservation functionalities, but not at the price of replacing their systems. Such an integrated solution may be the only feasible way to introduce PDS to existing archives.

3.5.2 Strategy

We aim to add to the existing system rather than replace it, taking advantage of the existing functionality, technology, and hardware, and providing the additional features needed for long term digital preservation. The existing interface should be enhanced to include preservation functionality and/or the PDS interface may be added as an additional interface. The objective is to add support for bit preservation and logical preservation to the existing system, as needed, to address preservation aware storage requirements (see 3.2).

3.5.3 Integration architecture

3.5.3.1 Basic architecture scheme

PDS can be integrated with existing file systems and archives to enhance the systems with support for OAIS-based long term digital preservation. Figure 10: *Integrating PDS with existing archives or file systems* below depicts the basic architecture scheme for such integration. We propose the addition of two components to the existing archive: an *AIP Generator* of manifest files and a *PDS box*. The *AIP Generator* generates AIPs, where each AIP is a manifest file with links to the existing content data and other metadata that already exists in the archive. If some metadata is missing (e.g., RepInfo) the AIP Generator is programmed to add the metadata by embedding it into the manifest file or as a separate file or database entry linked from the manifest file. Sometimes, programming the AIP Generator to generate the manifest files can be quite simple, for example, if there is an existing naming scheme that relates the various AIP parts.



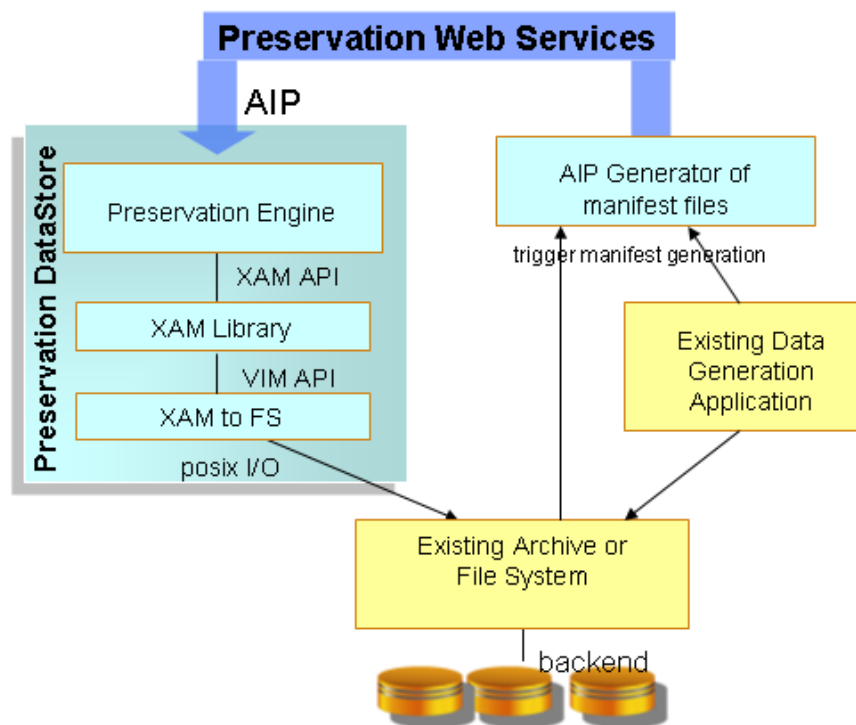


Figure 10: Integrating PDS with existing archives or file systems

The AIP Generator can be triggered to generate a manifest file either by the existing archive or by the existing Data Generation Application. The former method is possible if the existing archive has the mechanism to trigger an event when new content data is ingested into the archive. The latter method is possible if the existing data generation application has the mechanism to hook the AIP Generator when new content data is generated. Note that in both cases, data can be entered into the archive using the existing data generation applications and thus does not require writing new applications.

The AIP generator may be used to generate AIPs for data objects that are already stored in the existing storage system or for new ones. Triggering the AIP generator for old documents may add the ability to preserve them for long periods of time.

The generated manifest AIPs are ingested into the second component – the *PDS box*. Most of the PDS functionality is provided, including awareness of the AIP structure, execution of data-intensive functions such as transformations within the storage, handling technical provenance records internally, support for media migration, and maintenance of referential integrity. However, PDS cannot, of course, guarantee intelligent data placement. It cannot guarantee physical co-location of the various parts that constitute the AIP and, likewise, it cannot guarantee the aggregation of related AIPs into clusters placed on the same media unit. However, when the next periodic migration phase happens because of media decay or for other reasons, PDS can optimize the data placement and provide physical co-location of related AIPs.

The PDS box uses the existing system as its underlying storage and uses it to store its XAM objects. This requires mapping the XAM objects to the existing system, thus adding another component that implements the VIM API on top of the existing system. This new XAMto<ExistingArchive> component replaces the current XAMtoOSD component. The current OSD object layer is not integrated with the existing system.

To summarize the integration effort: the existing system and application remain unchanged, a new component - the AIP generator - is added to construct the OAIS compliant data objects, and a new mapping component binds the PDS box to the existing system.

The PDS interfaces may be exposed as part of the existing interface, or in addition, or both.





3.5.4 Variants

3.5.4.1 Utilizing XAM and OSD layers

While the utilization of the Preservation Engine layer in the integration architecture is essential, as it provides the long term preservation functionality, the utilization of the underlying layers, the XAM and OSD layers, is optional.

The general case presented in the previous section uses the Preservation Engine and XAM layers of PDS and replaces the OSD object layer with the existing storage system.

In some storage systems, adding the XAM layer on top of the existing system may add a high overhead along to its benefits.

An alternative is to map the preservation engine layer directly to the existing system, thus giving up the utilization of both the XAM and the object layers. Using this option requires binding the preservation engine layer to the existing system instead of the XAM layer. (See Figure 11: *Integrating PDS without the XAM mid-layer*).

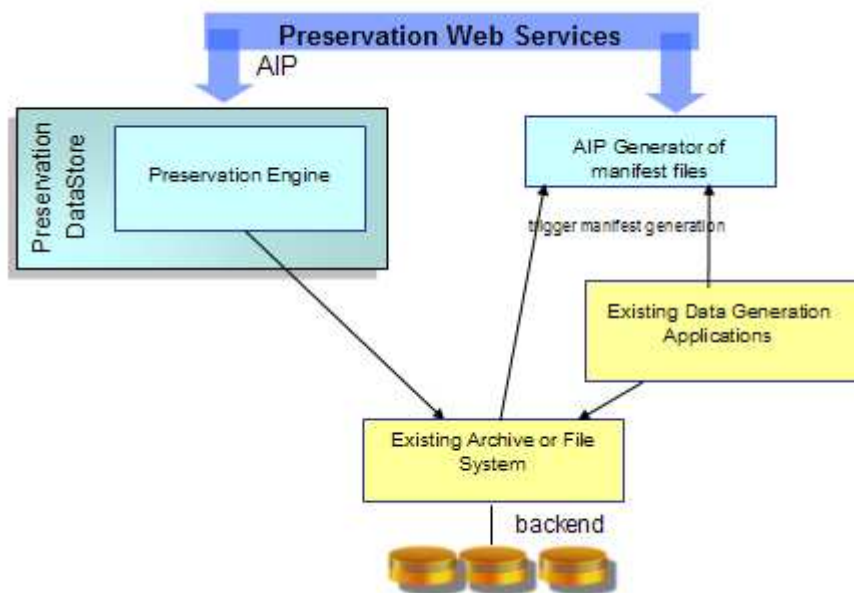


Figure 11: Integrating PDS without the XAM mid-layer

As mentioned before, binding the XAM layer to the existing system requires implementing the VIM API for this system, which is part of the XAM emerging standard.

PDS is built in such a way that the preservation engine layer is unaware of the underlying storage mid-layer. It uses a well defined interface called IAipToStorage that is currently implemented on top of XAM by the XamToAip component in PDS. Binding the preservation engine directly to the existing system requires implementing this interface for this system. The preservation engine layer does not require any internal modifications to support this mapping. A more detailed description of both mapping options is in 3.5.6.2.

In the previous variants we described options that use the existing storage system as the only storage system and store all data and metadata on it (both options use the existing storage system and do not use the OSD object layer of PDS). An alternative is to use the existing storage system only for the content data. The remaining parts of the AIP, i.e., the content data RepInfo and the PDI, are stored in the PDS backend storage, the OSD object layer, either while using XAM as a mid-layer or directly.





3.5.4.2 Encapsulation of components

The AIP generator component may be external to the PDS box, as was presented previously, or encapsulated in the PDS box.

If we use an external AIP generator, it may have a separate user interface, and may be expanded or replaced independently of the PDS box. This way the PDS box requires less changes for integration but the introduction of PDS into the existing system is more complex and expensive.

The alternative is to encapsulate the AIP generator in the PDS box. In this option the existing system is unaware of this additional component; it does not have a separate user interface as it is used as part of the PDS box. It is harder to enhance and configure, but the integration process is simpler and cheaper.

This simplicity may be valuable for initial integrations. See Figure 12: *Integrating PDS with encapsulated AIP generator and combined interface*.

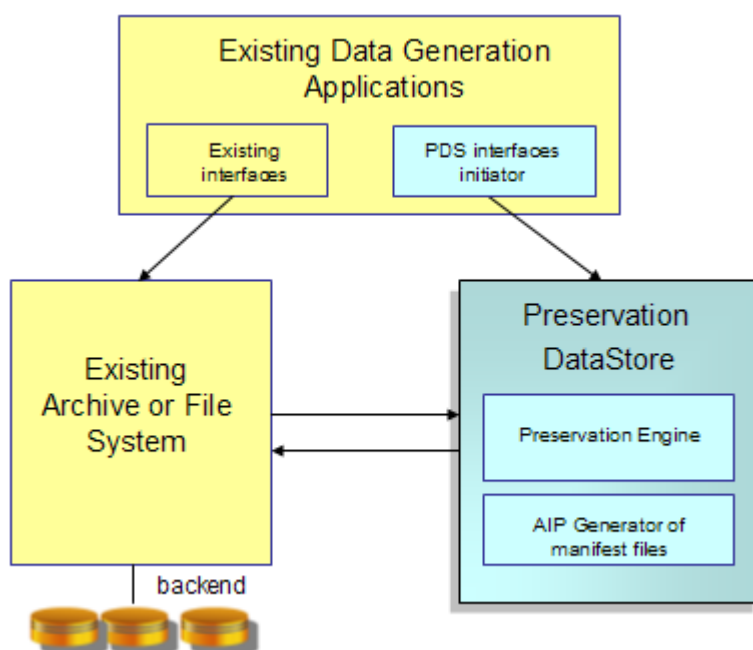


Figure 12: Integrating PDS with encapsulated AIP generator and combined interface

3.5.5 Interfaces

The PDS interface may be exposed independently from the existing interface or as part of it. Since it is preferable to have one coherent interface for a system, we aim to expose the PDS web service interface using the existing application, and forward the appropriate calls to the PDS box.

In addition, the existing interface may be expanded to include long term preservation options. For instance, within the existing ingest interface we may have an additional option to store the data object for the long term. This way the user may decide for each data object if it is meant for long term or short term preservation. If the user intends the data object for long term preservation, the existing interface redirects the request to PDS. It may first store the data object as usual, and then call the PDS API for ingesting stored data objects. This call to PDS may also be used when data objects that were stored in the past (before the integration) are now meant for long term preservation. This interface (ingestForStoredDocument) is optional for implementation in PDS but mandatory for a PDS box that is integrated in an existing system. See Figure 12: *Integrating PDS with encapsulated AIP generator and combined interface*.





3.5.6 New components

There are two main components that should be implemented when integrating PDS with an existing system. The AIP generator is responsible for compliance with the OAIS information model by creating AIPs for the content data objects that are stored in the existing system. This component is independent of the specific storage system and the same implementation may be used for different integrations.

The second component is the mapping component that binds PDS to the existing storage system. This component may need to be implemented separately for different integrations since it uses the specific features and functionality of the existing system. Also, different integration architectures require a different mapping component.

3.5.6.1 AIP Generator

The AIP Generator generates metadata-enriched OAIS AIPs from a document stored in an existing storage system. This is done without any additional input from the end-user. The AipGenerator class (see the UML class diagram in Figure 13: *UML Class Diagram for AIP Generator*) receives a reference to the stored document and creates an AIP whose Content Data Object (CDO) is a reference to the stored document, and whose RepInfo and PDI consist of metadata extracted from the CDO itself and from the document properties that are retrieved from the storage system. The metadata extracted by the AIP Generator can be structural (file size, application name, etc.) or semantic (document title, category, etc.).

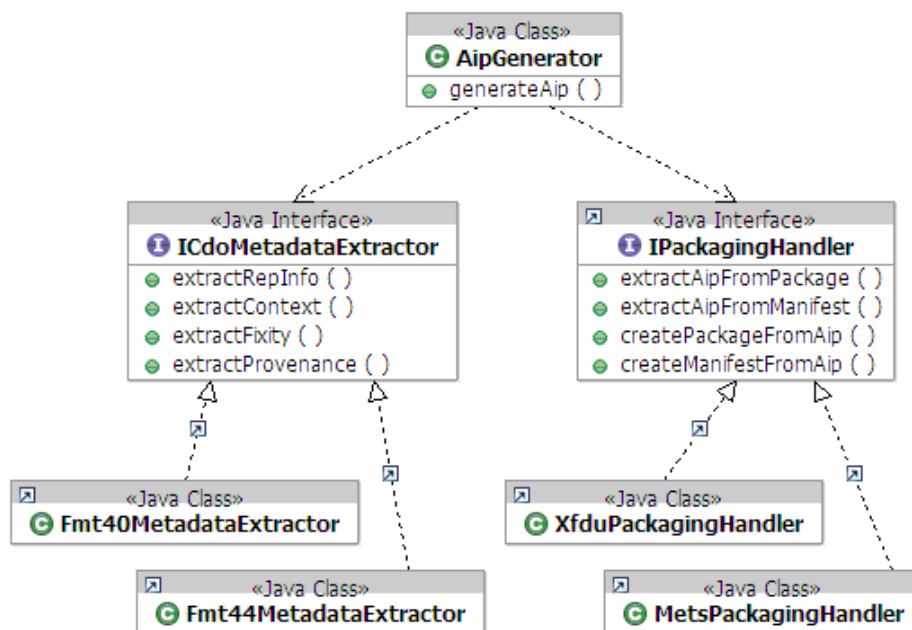


Figure 13: UML Class Diagram for AIP Generator

AIP Generator is agnostic to the existing storage system, to the AIP packaging format, and to the format of the stored file it handles: document properties are passed to AipGenerator in a data structure agnostic to the specific storage system; packaging of the AIP and extraction of the CDO metadata are done through interfaces – IPackagingHandler and ICDOMetadataExtractor, respectively – which are implemented for each format.

3.5.6.2 Mapping layer

The mapping layer is the layer that binds PDS to the existing storage system. As a result, it needs to be re-implemented on each integration to connect PDS to the specific storage system





and to use its unique functionality. Following are the two optional layers to bind to the existing system, depending on the integration architecture.

3.5.7 Mapping Preservation Engine

The following describes the direct mapping of the preservation engine layer to the existing system. This mapping should be implemented if the XAM layer is not utilized.

In this mapping, the manifest file that is generated by the AIP generator is stored as a single object in the existing system, unlike the mapping to XAM that allows storing each AIP sub-section separately. This is due to the likelihood that the existing storage system does not provide support for complex objects such as XAM provides. As a result, this mapping may complicate the support for access methods to each AIP sub-section and require parsing the manifest to implement those functions.

The AIP is mapped into two files: one for the content data and one for the manifest file. The manifest file contains references to the content data RepInfo objects, as they are stored as separate AIPs. The PDI sections are embedded. In addition it contains a reference to the content data object.

In this case, the mapping component implements the IStorageHandler interface that includes methods to open/close a session to the storage provider, create objects and read/write to them. See appendix for the complete interface.

Once this interface is implemented for the specific existing system, the preservation engine layer uses it transparently. In addition there is a need to implement the IStorageHandlerCreator interface, which is used to generate storage handlers by the storage handler factory in the preservation engine.

3.5.8 Mapping XAM

The AIP maps to XAM objects as it maps in the current design, taking advantage of XAM that provides complex objects that bundle together pieces of data and metadata. For the detailed mapping, see D2201, section 6.2.

Under the XAM library module lies the Vendor Interface Module (VIM), which acts as a bridge between the XAM standard APIs and the vendor storage systems. In this case the mapping component implements the VIM API (see the XAM specification for details) on top of the existing system.

The content data objects may be stored in the system before the AIP is generated and consequently also before the XAM objects are generated. Therefore we may not be able to place the content data object as part of the XAM object. The XAM object will need to keep a reference to it instead of having it embedded in the XSet, as in the current design.

The XAMtoAIP component, which provides the mapping of the preservation engine to XAM by implementing the IStorageHandler interface, needs to be aware that in some cases, instead of the actual content data, the XAM object merely contains a reference to it.

To reference the content data we put an external link in the content data section (usually a structure accompanied by structural RepInfo).

3.5.9 Integrating with ECM using CMIS

Integrating with an existing system using a standard protocol can make the mapping layer independent of the existing system with which PDS is integrated and therefore a single implementation enables easy integration of PDS.

The Content Management Interoperability Services (CMIS) standard is a uniform means for applications to work with content repositories. The CMIS standard is part of IBM Business Content Services integrated approach to help you manage and control content created by shared





workgroups across your organization. By participating in the development of CMIS, IBM is actively driving better interoperability between Enterprise Content Management (ECM) products regardless of vendor and providing clients with more choice and lower costs when it comes to basic ECM needs.

More than a year in development, the CMIS standard is supported by IBM, Microsoft, SAP, BEA/Oracle, EMC, OpenText and Alfresco. Submission of the standards and specifications to the Organization for the Advancement of Structured Information Standards (OASIS) is now underway and is anticipated to be approved in 2009.

If the existing system is an ECM that supports CMIS the integration work may be simpler and the mapping component may be reused.

3.6 EXAMPLES

3.6.1 Integrating PDS with iRODS

3.6.1.1 SRB/iRODS overview

The Storage Resource Broker (SRB)/ Intelligent Rule-Oriented Data management System (iRODS) [9, 10, 11, 12] is data grid technology developed and owned by the San Diego Supercomputing Center (SDSC). It manages distributed data, enabling the creation of data grids that focus on the sharing of data, and was recently extended to *persistent* archives that focus on the preservation of data. Data grid technology provides fundamental management mechanisms for distributed data in a scaleable manner.

This includes support for managing data on remote storage systems, a uniform name space for referencing the data, a catalog for managing information about the data, and mechanisms for interfacing with the preferred access method. The SRB/iRODS is middleware software; it builds on top of standard file systems, commercial archives and storage systems.

The basic architecture of SRB consists of an SRB server and a Metadata Catalog (MCAT) server. The SRB server exposes various file-like APIs to the application and interacts with the storage system. The MCAT server handles the information stored in the SRB database. The iRODS adds to it a rule-based system (see Figure 14: ***SRB/iRODS system***), which allows the user to make assertions about the rules under which the data is being maintained.



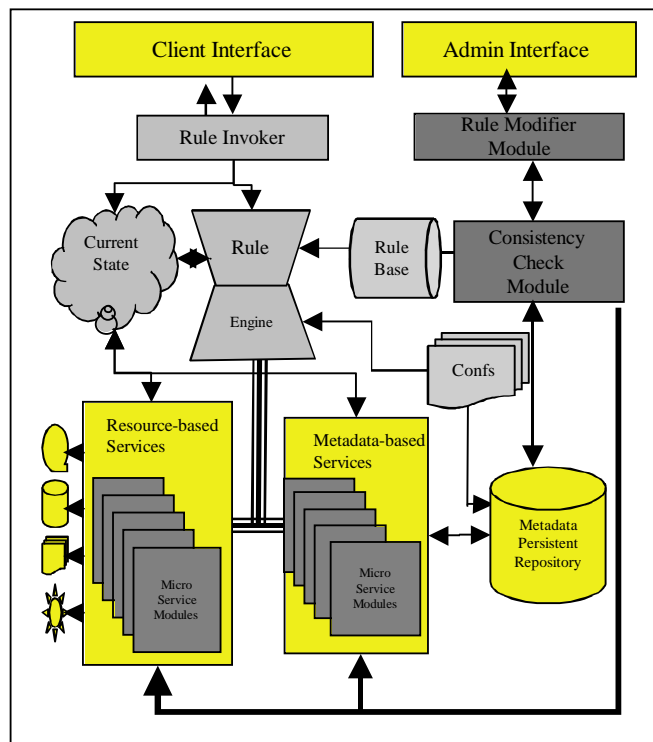


Figure 14: SRB/iRODS system

3.6.2 Integration architecture with SRB/iRODS

PDS layers can be integrated with SRB/iRODS both on the top and the bottom (see Figure 15: *PDS and SRB/iRODS utilizing XAM and OSD as storage interfaces*).

The preservation engine layer can extend today's SRB/iRODS application interface layer and expose the PDS API. iRODS provides a Java I/O library interface called Jargon that may make it easy to port the PDS API on top of iRODS.

The XAM library may be exposed as an application interface (on the top) or as a storage interface (on the bottom).

The XAM VIM should be placed at the storage interface layer. If we use OSD as the backend storage, the PDS XAMtoOSD VIM component may be utilized. Otherwise, another VIM implementation is required. The OSD layer may be placed at the storage interface layer. The utilization of XAM and OSD layers is optional. As described in the previous chapter, not utilizing the XAM library will require implementing a mapping layer of the preservation engine to iRODS. Utilizing XAM without the XAMtoOSD component requires an implementation of VIM on top of different backend storage.



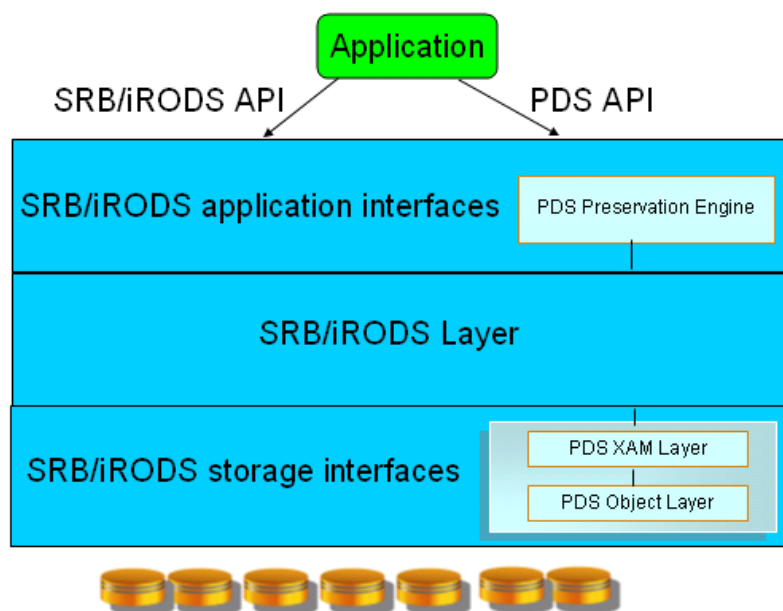


Figure 15: PDS and SRB/iRODS utilizing XAM and OSD as storage interfaces

3.6.3 Preservation engine integration design points

3.6.3.1 Interfaces

iRODS supports multiple types of APIs. They range from web browsers to workflow systems, to Fedora digital library middleware and to Unix shell commands. A port of the DSpace digital library is expected on top of iRODS. Both DSpace and Fedora provide a set of preservation related services.

PDS API may be provided as one of the mechanisms for accessing iRODS that supports building and manipulating AIPs. Such AIPs may be generated by the user and possibly packaged in XFDU or SAFE formats, or by the PDS AIP generator on behalf of data objects intended for long term preservation.

iRODS micro-services make it possible to apply the PDS API directly on files at remote storage locations.

3.6.3.2 AIP support

The knowledge for manipulation of a particular form of an AIP can reside in the client, in structured information drivers, or in micro-services.

The first approach is to have the AIP creation performed by a client that then stores the result within an iRODS data grid. This is done by DSpace and Fedora.

iRODS supports the concept of "structured information". This consists of the mechanisms required to access an information resource, query the resource, and extract the information needed for further interactions. This mechanism requires a dedicated driver that implements the "access protocol" of the information resource. This is currently supported for HDF5, tar files, and is planned for SQLite.

An alternative is to encapsulate the AIP mechanisms in micro-services that are then applied by iRODS at the remote storage location. This approach enables the creation of a server-side workflow that chains together multiple operations such as extraction of an XML schema from an AIP, comparison with a template schema for presence of required metadata, and execution of recovery mechanisms when discrepancies are found.





Integrating PDS into iRODS encapsulates the AIP support into the application interface and exposes it in the PDS API.

3.6.3.3 Bit preservation and data migrations

iRODS supports bit preservation. It manages checksums, replicas, and synchronization of files. SRB handles media failures, data mirroring, and distribution of data. iRODS manages a logical name space that can be used to impose a collection hierarchy. iRODS tracks the mapping between the logical name and the actual physical location as files are moved to new storage locations.

PDS needs to document these events in the provenance section of the AIPs.

3.6.3.4 Data transformations

iRODS supports the remote execution of procedures. Procedures are composed from sets of micro-services that can be used to parse data formats, convert to new data formats, or extract metadata. Micro-services can be used by PDS to provide data transformation services via the PDS API.

3.6.3.5 Metadata handling

Fixity computations may be handled by iRODS as it manages checksums on data files. Metadata can be extracted into an XML file, and then iRODS can manage checksums on the XML file and thus handle fixity computations on metadata. iRODS also manages audit trails that track all operations performed on a file.

This provides some of the metadata handling needed for long term preservation. PDS can add API to document provenance events by the user, and update the PDI section of an AIP with documentation on provenance and fixity events.

3.6.3.6 Physical co-location of an AIP

As described above, metadata can be stored in a file. iRODS supports a logical arrangement (mapping into a logical name space iRODS controls), physical placement of files into selected remote storage locations, and packaging (aggregation of files in a single file).

PDS can use these capabilities to co-locate each AIP and related AIPs on the same media.

3.6.3.7 Policies

The preservation policies may be moved into rules and procedures that are automatically enforced by the data management system, independently of the choice of API.

The rule mechanism may be also used to introduce preservation policies for PDS. The set of preservation policies that need to be enforced may be dynamically changeable.

3.7 STORAGEHANDLER INTERFACE

```
// begin_generated_IBM_prolog
//
// prolog.pro.ococ.tap
//
*****
// IBM Confidential
//
// OCO Source Materials
//
// (C)Copyright IBM corp. 2007
//
// The source code for this program is not published or otherwise divested of
```





```

// its trade secrets, irrespective of what has been deposited with the US
// Copyright Office.
//
//
*****
//
// end_generated_IBM_prolog

public interface IAipToStorage {

    /**
     * Initializes the storage handler.
     * @param <args> Arguments for the initialize method
     * @return void
     */
    public void initializeStorageHandler(String args[]);

    /**
     * Initializes the storage provider.
     * @param <args> Arguments for the initialize method
     * @return void
     * @exception StorageException
     */
    public void initializeStorage(String args[]) throws StorageException;

    /**
     * Closes the storage provider
     * @return void
     * @exception StorageException
     */
    public void closeStorage() throws StorageException;

    /**
     * Opens a session with the storage entity.
     * @return void
     * @exception StorageException
     */
    public void openSession() throws StorageException;

    /**
     * Closes the session with the storage entity.
     * @return void
     * @exception StorageException
     */
    public void closeSession() throws StorageException;

    /**
     * Creates a container object to store the AIP in.
     * @param <pAip> The AIP to be stored
     * @param <pId> The AIPID to be stored
     * @return void
     * @exception StorageException
     * @throws PDSIllegalParameterException
     */
    public void createContainer(PDSAip pAip, byte[] pId) throws
StorageException,
PDSIllegalParameterException;

    /**
     * Opens the container object the specified AIP is stored.
     * @param <pAipId> The ID of the AIP
     * @param <pContId> Container ID
     * @param <modify> true/false
     * @return void

```





```

    * @exception StorageException
    * @throws PDSIllegalParameterException
    */
    public void openContainer(byte[] pAipId, byte[] pContId, boolean pModify)
throws
                                StorageException,
PDSIllegalParameterException;

    /**
    * Closes the container.
    * @return void
    * @exception StorageException
    */
    public void closeContainer() throws StorageException;

    /**
    * Abandons the container.
    * @return void
    * @exception StorageException
    */
    public void abandonContainer() throws StorageException;

    /**
    * Gets the length of the specified piece of data.
    * @param <pName> Data name
    * @return Length of data
    * @exception StorageException
    */
    public long getDataLen(String pName) throws StorageException;

    /**
    * Commits the transaction.
    * @return Byte array containing the ID
    * @exception StorageException
    */
    public byte[] commit() throws StorageException;

    /**
    * Writes the Content data to the appropriate place in the container.
    * <p>If there is content data already written this method will write over the
    * existing data.</p>
    * @param <pName> Content data name
    * @param <pDataOut> A byte array containing the Content data
    * @param <pLength> Length of the Content data
    * @return void
    * @exception StorageException
    */
    public void writeContentData(String pName, byte pDataOut[], int pLength)
throws
StorageException;

    /**
    * Reads the Content data from the appropriate place in the container.
    * @param <pName> Content data name
    * @param <pDataOut> A byte array to contain the Content data
    * @param <pLength> Length of the Content data
    * @return void
    * @exception StorageException
    */
    public void readContentData(String pName, byte pDataIn[], int pLength)
throws
StorageException;

    /**

```





```

    * Writes PDI data to the appropriate place in the container.
    * @param <pName> PDI section data name
    * @param <pRecords> Array of PDSpdiRecords to store
    * @return void
    * @exception StorageException
    */
    public void writePdiSectionData(String pName, PDSpdiRecord[] pRecords)
throws
StorageException;

    /**
    * Reads PDI data from the appropriate place in the container.
    * @param <pName> PDI section data name
    * @return Array of PDSpdiRecords stored
    * @exception StorageException
    */
    public PDSpdiRecord[] readPdiSectionData(String pName) throws
StorageException;

    /**
    * Look up the corresponding ID and add it as a property.
    * @param <pName> RepInfo ID field name
    * @param <pRepInfoId> AipId of RepInfo
    * @return void
    * @exception StorageException
    */
    public void writeRepInfo(String pName, PDSRepInfo pRepInfo) throws
StorageException;

    /**
    * Read the ID and look up the corresponding AIP ID.
    * @param <pName> RepInfo ID field name
    * @return <pRepInfoId> AipId of RepInfo
    * @exception StorageException
    */
    public PDSRepInfoRecord[] readRepInfo(String pName) throws
StorageException;

    /**
    * Write the properties in the Data Management entity.
    * @param <pProperties> a structure with data to be written to the Data
    * Management entity
    * @return void
    * @exception StorageException
    */
    public void writeProperties(Properties pProperties) throws
StorageException;

    /**
    * Read the properties from the Data Management entity.
    * @return <Properties> a structure with data read from the Data Management
    * entity
    * @exception StorageException
    */
    public Properties readProperties() throws StorageException;
}

```

3.8 PDS CONCLUSIONS





PDS is an archival storage system that may be deployed as a stand-alone system. However, a more likely case is that an archive that is required to support long term digital preservation will be integrated with PDS. This integration will include some parts of the current PDS implementation and some additional components.

PDS was designed from the outset to enable easy replacement of each of its layers: the Preservation Engine, the XAM layer, and the OSD layer. Each layer implements a well defined interface and is independent of the underlying layers.

This document shows the basic architecture and design for such integration. Any concrete integration requires a specific scheme of architecture and implementation of additional components to bind PDS to the existing system and provide long term preservation functionality for the data that was stored in the system in the past and for future data.

In the context of CASPAR, it would be useful to work on such an integration design with CASPAR data holders who wish to add long term digital preservation support to their existing storage systems.

To that end, PDS was installed in ESA and ASemantics.

3.9 PDS REFERENCES

ISO 14721:2003, Blue Book. Issue 1. CCSDS, 650.0-B-1: Reference Model for an Open Archival, Information System (OAIS), (2002)

"Towards OAIS-Based Preservation Aware Storage - A White Paper". See <http://www.haifa.il.ibm.com/projects/storage/datastores/public.html>

M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef, and J. Satran. "The Need for Preservation Aware Storage - A Position Paper". ACM SIGOPS Operating Systems Review, Special Issue on File and Storage Systems, Volume 41, Issue 1, pages 19-23, (2007)

M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef, J. Satran, and D.L Giaretta. "Preservation DataStores: Architecture for Preservation Aware Storage", In Proc. IEEE Conference on Mass Storage Systems and Technologies (MSST), San Diego, USA (2007)

SNIA - Networking Industry Association, Data Management Group, XAM (Extensible Access Method). See <http://www.snia-dmf.org/xam/>

Michael Factor, Kalman Meth, Dalit Naor, Ohad Rodeh, and Julian Satran. Object storage: The future building block for storage systems. A position paper. In Local to Global Data Interoperability – Challenges and Technologies, Sardinia Italy., pages 119–123, June 2005.

SNIA - Storage Networking Industry Association. OSD: Object Based Storage Devices Technical Work Group

International Committee for Information Technology Standards (formerly NCITS), SCSI Object-Based Storage Device Commands (OSD). Document Number: ANSI/INCITS 400-2004, technical editor: R.O. Weber, December 2004

Reagan W. Moore and Richard Marciano. "Building preservation environments". In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 424–424, New York, NY, USA, 2005. ACM Press

Reagan W. Moore, Joseph F. JaJa, and Robert Chadduck. "Mitigating risk of data loss in preservation environments". In *MSST '05: 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)*, pages 39–48, Washington, DC, USA, 2005. IEEE Computer Society

SRB – Storage Resource Broker. See http://www.sdsc.edu/srb/index.php/Main_Page

iRODS - Intelligent Rule-Oriented Data management System. See http://irods.sdsc.edu/index.php/Main_Page





4 THE ESA DEVELOPMENT

The ESA implementation plan implements two strategies:

Development of a 100% CASPAR-based system prototype to validate the CASPAR preservation features in view of a possible future integration of some components into the ESA infrastructure

Integration of CASPAR with another existing project that provides to users open access to the **ESA** and other Earth Science **archives** through the same **interface**

4.1 POSSIBILITY OF APPLYING THE CASPAR FEATURES TO ESA (AND OTHER) DATA

Currently, data (which come from sensors on different platforms, such as satellites, planes, boats, balloons, or located at ground on the land), information about the state of the Earth, relevant services, analysis results, applications and tools are accessible in a very scattered and uncoordinated way.

ES community would significantly benefit from a shared global ES infrastructure able to give simple access to historical data holdings and networks of sensors, broadband communications via ground and space, efficient, effective and distributed computing and storage resources, etc. In the context, ESA is leading the GENESI-DR project, which will develop an ES dedicated infrastructure providing reliable, easy, long-term access to Earth Science data and services via the Internet.

- With regards to data curation, the prospect of losing the digital records of science (and with the specific unique data, information and publications managed by ESA) is very alarming. To respond to the urgent need for a coordinated and coherent approach for the long term preservation of the existing European EO space data, ESA formed a Long Term Data Preservation (LTDP) Working Group.
- In addition ESA-ESRIN is participating to a number of international projects partially funded by the European Commission and concerned with technology development and integration in the areas of long term data preservation, such as CASPAR.

It should be clear from the previous analysis that there is a strong need of easily discovering and accessing data as well as adopting proper and effective data curation and preservation strategies. In this paper we report the results of the collaboration between CASPAR and GENESI-DR projects, addressing data curation and data discovery, access, and processing respectively.

This collaboration has the overall objectives of:

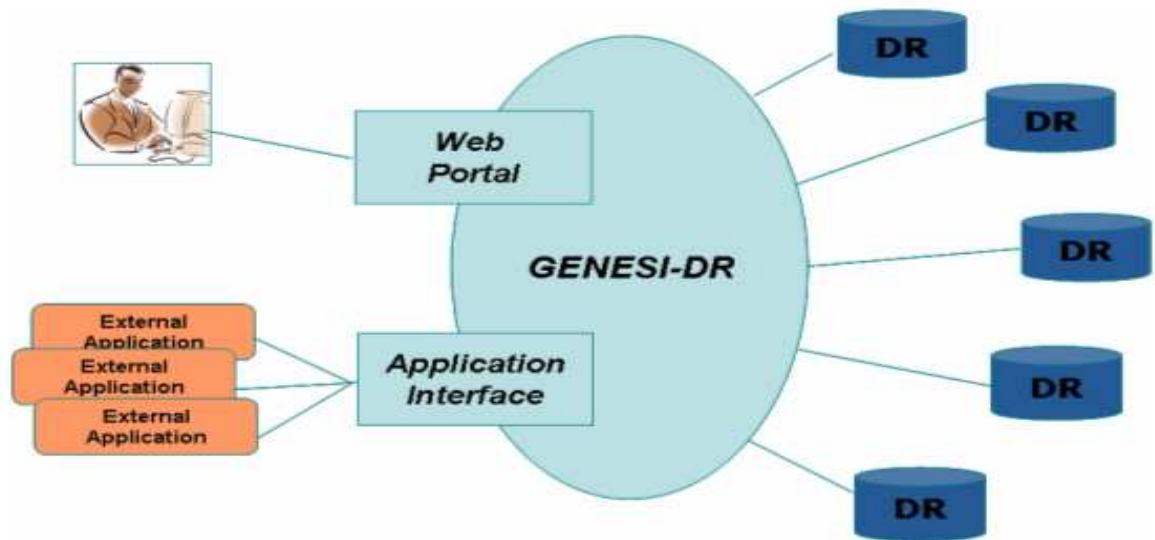
evaluating the effectiveness of the CASPAR data preservation framework as well as the impact on users and data providers while increasing its visibility in the Earth Science community by taking advantage of the GENESI-DR audience;

demonstrating the ability of GENESI-DR Research Infrastructure to provide easy access to heterogeneous and disperse data which are preserved according to the mechanisms defined in CASPAR.

4.2 THE GENESI-DR PROJECT

GENESI-DR (*Ground European Network for Earth Science Interoperations – Digital Repositories*) provides to users/applications open access to different European Earth Science Digital Repositories through the same interface.





Ground European Network for Earth Science Interoperations - Digital Repositories (GENESI-DR), an ESA-led, European Commission (EC)-funded two-year project, kicked-off early 2008 and is taking the lead in providing reliable, easy, long-term access to Earth Science data via the Internet. As previously discussed, Petabytes of data about our planet are available but distributed at different locations.

Currently, information about the state of the Earth, relevant services, analysis results, applications and tools are accessible in a very scattered and uncoordinated way, often through individual initiatives from Earth Observation mission operators, scientific institutes dealing with ground measurements, service companies, data catalogues, etc.

Data access is a major logistic problem. The EC has funded GENESI-DR as a flagship project in Europe to help meet the challenge of facilitating life of scientists from different Earth Science disciplines located across Europe in discovery, access and use (combining, integrating, processing, ...) of historical and fresh Earth-related data from space, airborne and in-situ sensors archived in large distributed repositories.

GENESI-DR is a response to the need for science users to be provided with data and tools to access, combine, and integrate the Earth-related data for performing their analyses.

This need has led to the identification of the basic GENESI-DR infrastructure requirements:

Capability, for Earth Science users, to discover data from different European Earth Science Digital Repositories through the same interface in a transparent and homogeneous way;

Easiness and speed of access to large volumes of coherently maintained distributed data in an effective and timely way;

Capability, for DR owners, to easily make available their data to a significantly increased audience with no need to duplicate them in a different storage system.

The first requirement is reflected in the GENESI-DR architecture on the Central Discovery Service, which allows users and applications to query information about data collections and products existing in heterogeneous catalogues, at federated DR sites.

This service can be accessed by users via web interface, the GENESI-DR Web Portal, or by external applications via open standardized interfaces exposed by the system.

More in detail, the Central Discovery Service identifies the DRs providing products complying with the user search criteria and returns the corresponding access points to the requester. This latter can refine its search towards the DRs, so that products complying with refined search





criteria are identified and the corresponding metadata are returned to the client. These include the product access URL allowing product retrieval.

To meet the second requirement, flexibility and performance are taken into consideration by making use of different and efficient data transfer technologies such as HTTPS, GridFTP and BitTorrent. To cope with the third requirement, the Architecture of GENESI-DR provides the DR owners with a mechanism (Catalogue Generator) to produce a metadata catalogue by simply harvesting metadata from their storage systems.

The Central Discovery Service communicates with the different DR catalogues through a web service gateway.

GENESI-DR platform is in line with the platform envisioned by DEGREE and with the expected achievements of the major ES communities programmes/initiatives, like GEOSS and INSPIRE. GENESI-DR has formal relation with GEO in several tasks such as: the management of large volumes and diverse types of Earth observation data; the implementation of the GEOSS architecture; the harmonisation of data, metadata and products; the use of satellites for risk management (inherited from the GPOD Fast Access to Imagery for Rapid Exploitation service).

GENESI-DR is also analysing common approaches to preserve the historical archives and the ability to access the derived user information as both software and hardware transformations occur. Ensuring access to Earth Science data for future generations is of utmost importance because it allows for the continuity of knowledge generation improvement.

4.3 THE CASPAR AND GENESI-DR COMBINED APPROACH

Recently a Working Group led by ESA has been created to establish a framework of collaboration between CASPAR and GENESI-DR in order 1) to adopt data preservation and curation mechanisms defined in CASPAR within GENESI-DR Research Infrastructure and 2) to provide CASPAR with a further way for validating the data preservation and curation mechanism in the domain of Earth Science, benefiting from GENESI-DR feedback and integrating services shared with other DRs to enlarge its capabilities to meet the ES community requirements.

To this end, different validation scenarios have been identified. The first step of this collaboration has been to define in detail the integration scenario between GENESI-DR and CASPAR. Two subsequent integration phases have been identified.

In the first phase the following actions have been performed:

GENESI-fication of a CASPAR based DR. A DR, in order to make its data discoverable and accessible through GENESI-DR, needs to perform the so called GENESI-fication procedure (described in detail in the GENESI-fication Guide). This action allows users to discover and access data that are preserved according the mechanisms defined in CASPAR.

Development of a validation service that estimates vertical profiles of Ozone starting from GOME L1 data stored in a CASPAR based DR. In this scenario, the user can discover and select, via GENESI-DR WebPortal, GOME L1 data (retrieved from the CASPAR based DR and preserved according to the mechanisms defined by CASPAR) and demand their processing, using GENESI-DR processors and computing resources, to obtain an estimate of vertical profiles of ozone.

Development of a validation service that generates GOME L1C data starting from GOME L1 data.

In this scenario, the user can discover and select, via GENESI-DR WebPortal, GOME L1C data. These data are generate on-the-fly using related GOME L1 and processors stored (and so preserved) in a CASPAR based DR.

These scenarios demonstrate:



how CASPAR can benefit from GENESI-DR in using shared services for discovering, accessing and processing preserved data;

how GENESI-DR can benefit from CASPAR in providing the user with preserved data and processors;

the possibility for CASPAR users to discover and access data generated on-the-fly and in a transparent way by using services provided by GENESI-DR and source data and processors specified by the curation mechanism.

In Figure 16, illustrating one of the validation scenarios previously described, the user obtains ozone profile information using the GENESI-DR processing software and the GOME L1 data preserved in CASPAR.

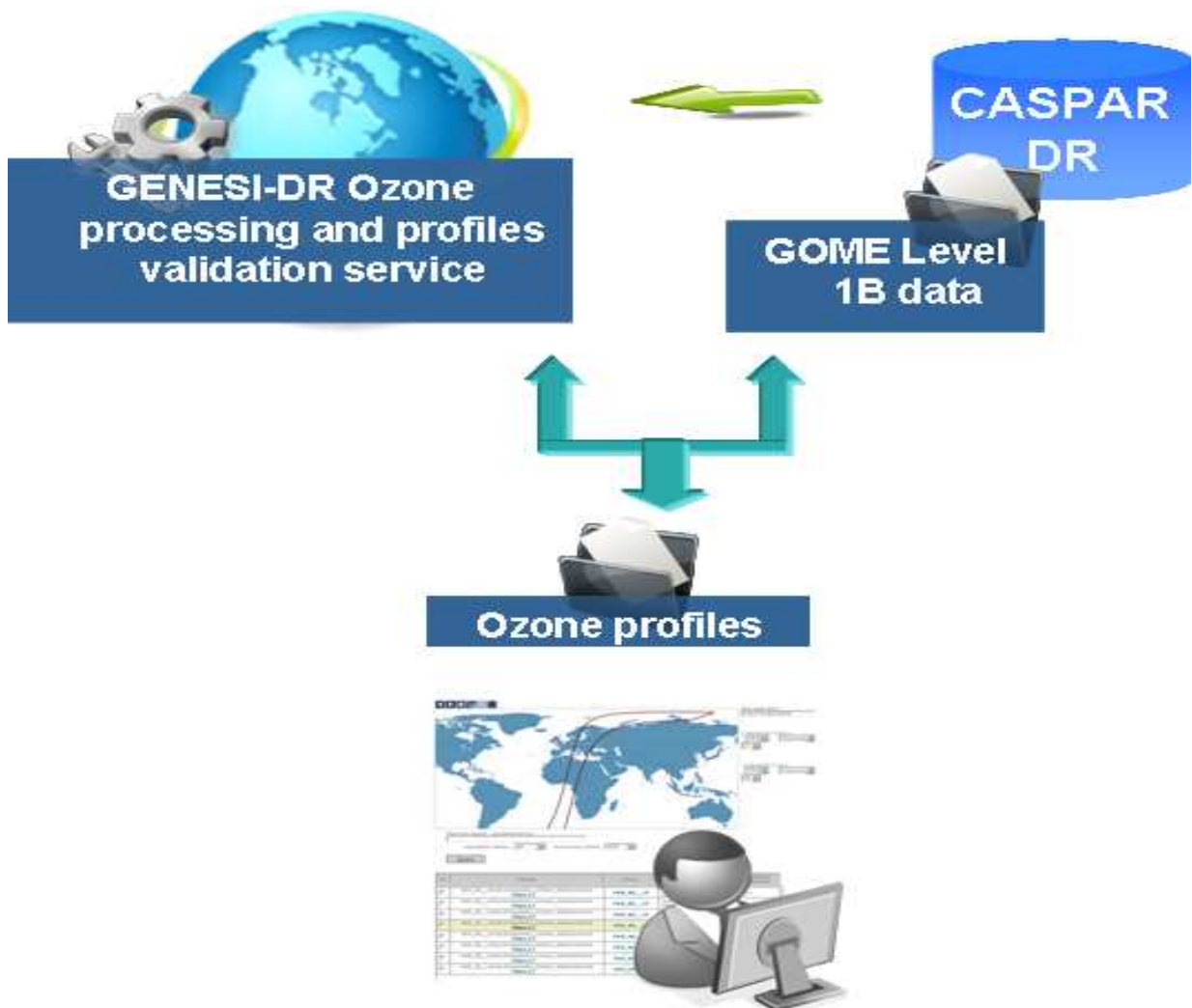


Figure 16 – Access and exploitation of preserved data

In Figure 17, illustrating another validation scenarios previously described, the user derives GOME LIC products from GOME L1 using processing software and data both preserved in CASPAR.



Figure 17 Access and exploitation of preserved data and preserved processors

In a second phase of the cooperation, the actions needed for the following tasks will be analysed and performed:

to make the previously defined services accessible from outside GENESI-DR webportal (e.g. webservices) ;

to store the processing results in CASPAR-DR;

to return profile-based RepInfo to GENESI users (see Figure 18: a less expert user asking for L1 products gets the products, processing software and related documentation, while an expert user performing the same query only gets products and related processing software);

to define a strategy for propagating CASPAR features to other relevant GENESI DRs.

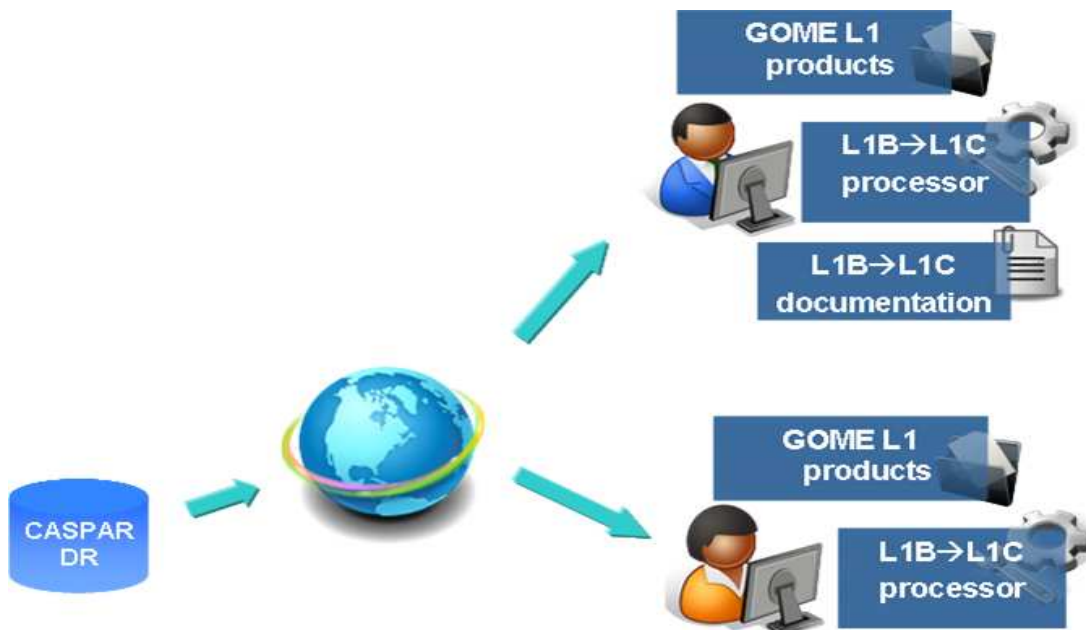


Figure 18 – Different Representation Information are returned to different users

4.4 STUDIES CONCERNING THE INTEGRATION BETWEEN CASPAR & GRID

4.4.1 The description of actual processing system:

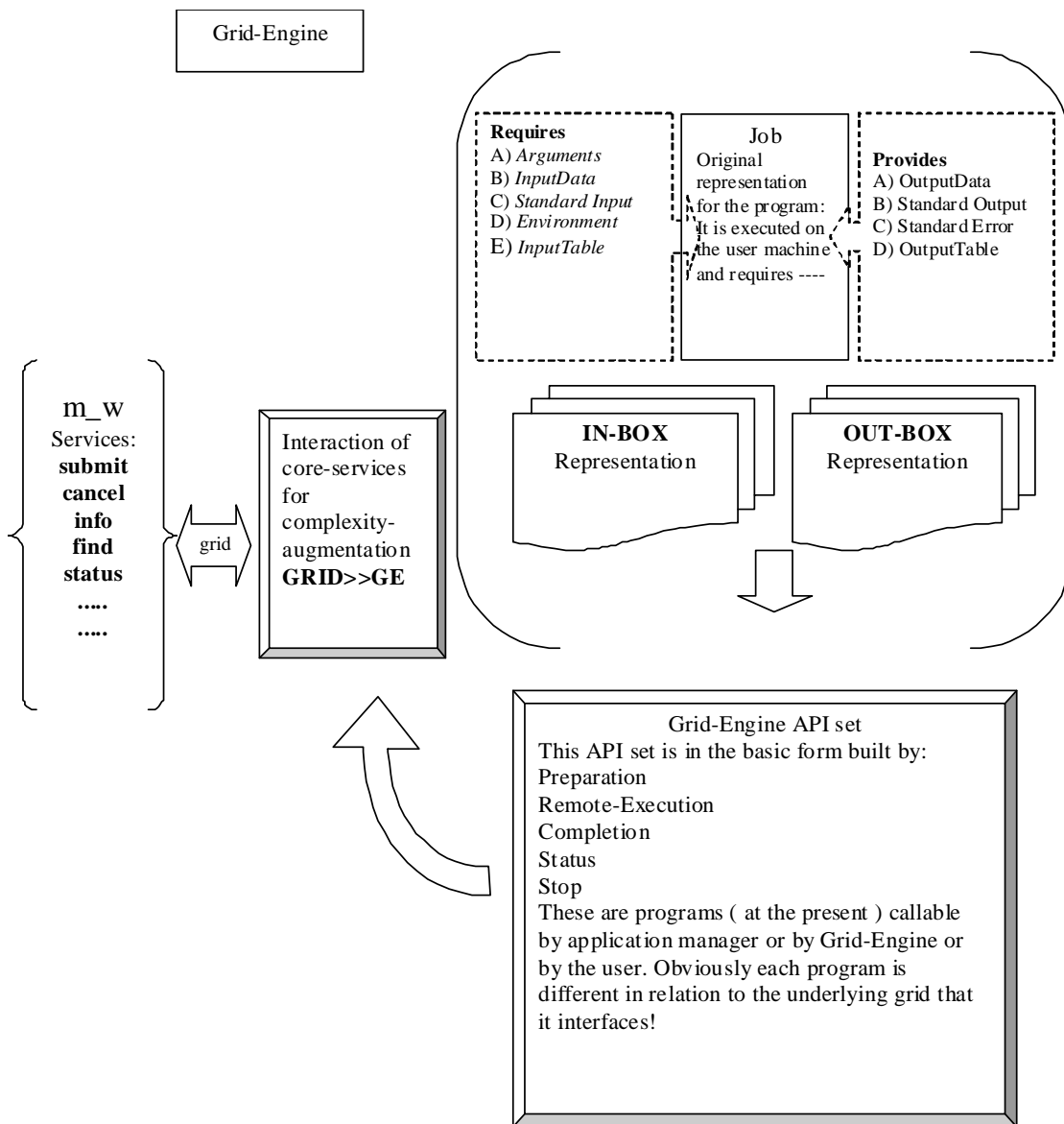
Grid-Engine is the WorkFlow Orchestrator for the EGEE Lite Grid, 100% Java.

The idea leading the Grid (called local to distinguish it from a real Grid) is the adoption of an API set interacting directly with the underlying middleware; in this way it is possible to allow an user (a scientist) to put his program on the “GRID”.

The grid-ification of a program is a complex operation so come out the concept of the application manager that is responsible for the migration of the job. The research focuses on the integration of the Meta-Data catalogs with the Ontologies/Dictionaries enabled by CASPAR.

Knowing that the first attempt of the integration between CASPAR & ESA_Grid is planned to be performed with GOME data;





The application manager makes 3 principal (logical and implemented) programs

- Preparation
- Wrapper
- Completion

For a middleware the difference between several Preparation is minimal only related to the name of In-Box and to the different internal call for fine-tuned utility script.

4.4.1.1 Light Job Description

Local (or **Locally**) indicates an action (actions) or file (files) on the machine that runs the Grid-Engine, while **Remote** (or **Remotely**) indicates that the inherent action (actions) or file (files) is to be considered on the CE-machine.

A better definition for the In and Out DATA:

This piece collects the **OUT-BOX** defined above and made by :
 Standard Output of the process invoked by the user (redirected on the stdOut file)





Standard Error of the process invoked by the user (redirected on the stderr file)

OutputData-set produced by the user-process [the files really produced by application probably remain on the Grid, in the storage-Places]

OutputTable [listing as URLs the whole OutputData-set]

The In DATA collects the **IN-BOX**,

Arguments (a text file containing the argv[] : argData)

StandardInput (a file containing all the informations required by the application ; we used input.in file)

InputData-set (The complete set of files to be elaborated , resident somewhere in the Grid)

InputTable [listing as LogicalFiles the whole InputData-set]

Environment: a set of variable-value pair

4.4.1.2 Preparation

The *Preparation* makes operations involved in the In_Data collection. It prepares the files; argData and input.in [checking if the InputData set is correctly available by the algorithm during execution] together with environment file. An example of *Preparation program* is :

Start

creates the input file

set Parameters

.... other here

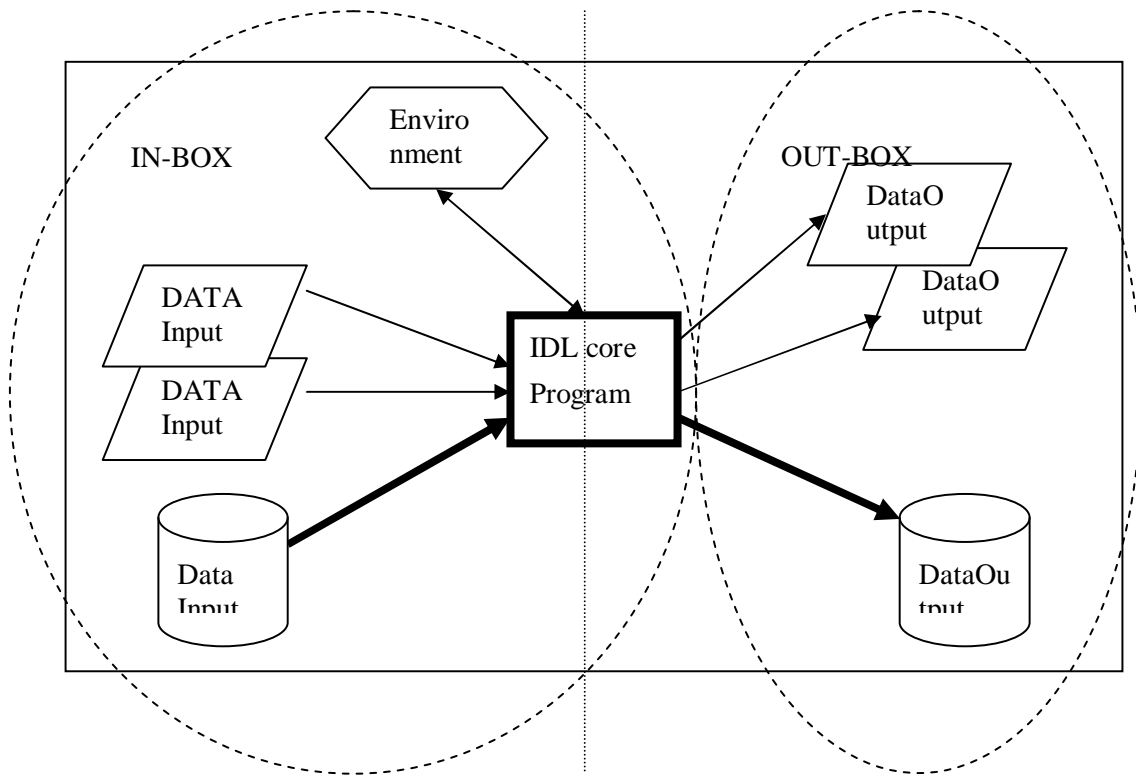
End

4.4.2 Remote Execution Phase

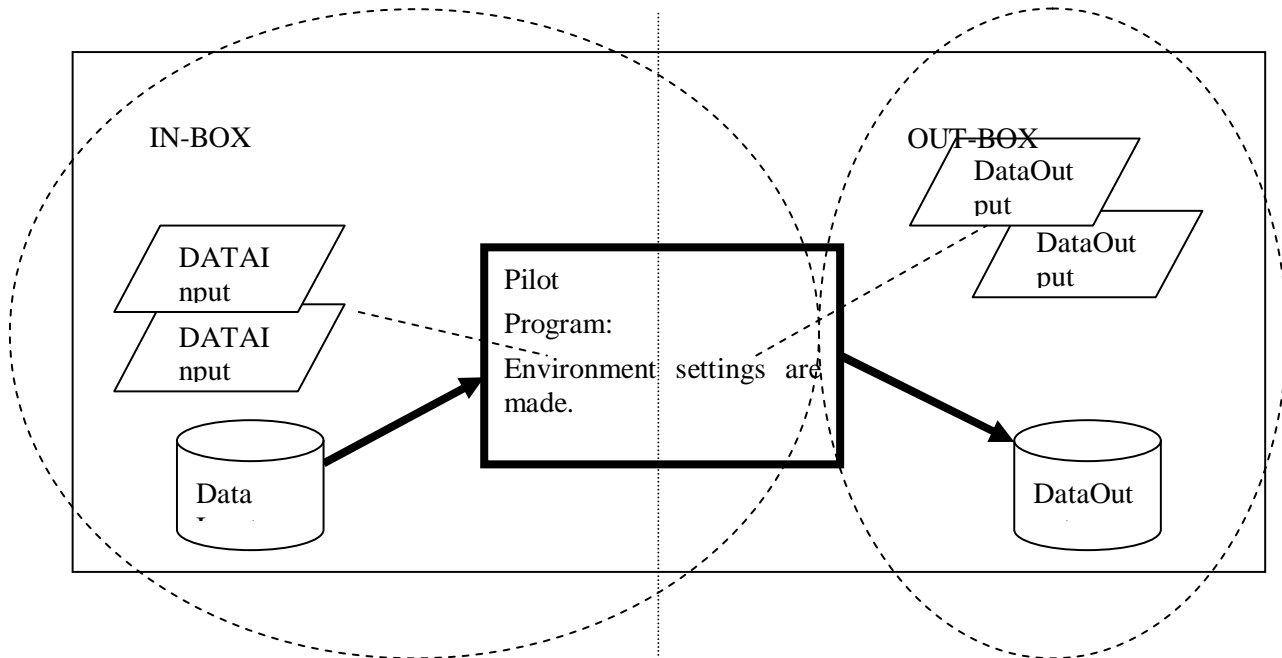
4.4.2.1.1.1 Store&Register procedure

The pilot script is the wrapper for the job (executed remotely): its name could be fixed at once or changed each time. Suppose that application manager has an IDL [r_Sinc] program that elaborates data stored in a file for a result production:





The translation to Grid-Engine enabled JOB of the previous IDL program consists in several operations.



Pilot Program is built regarding the informations related to IN-BOX and OUT-BOX.

The end of the preparation-section is the beginning of the wrapper-section.

It is clear that in the wrapper phase the Pilot Program runs as on the local machine and therefore it has to simulate the execution of the computational-algorithm but in a driven-automatic way, without any client or external-system interaction. For the computational-process launch on the local machine the client must furnish to the application these entities :





Arguments

InputData

InputTable

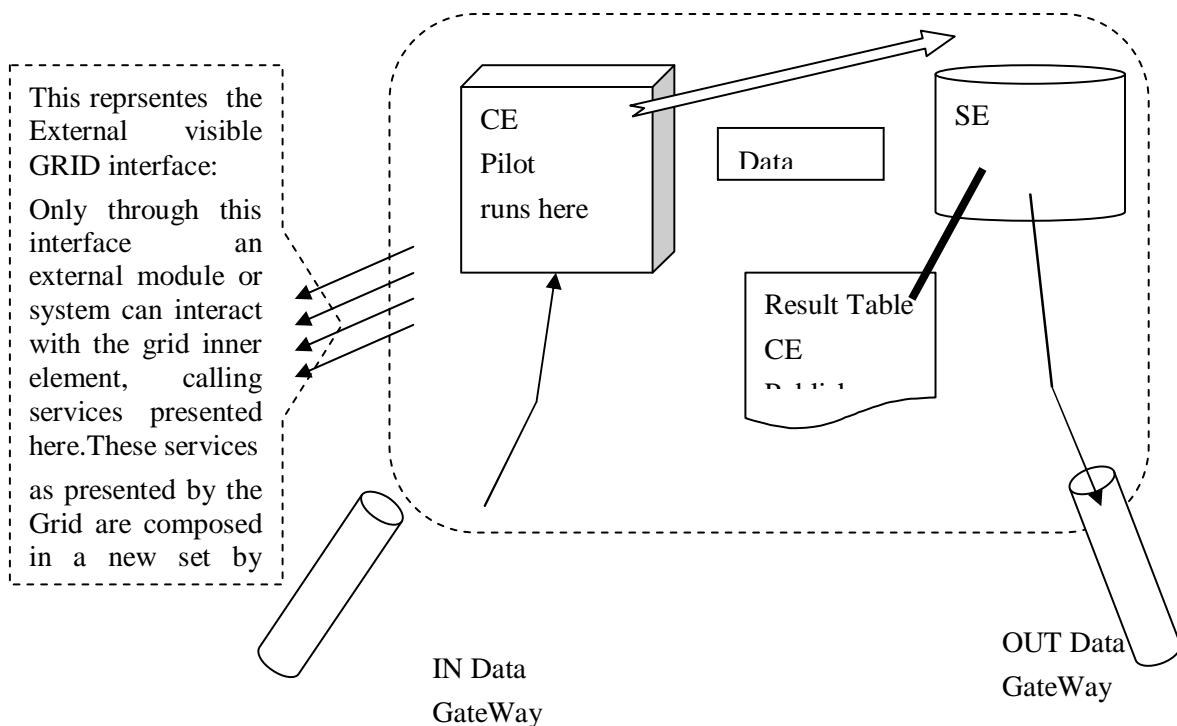
StandardInput

Environment

where StandardInput , Arguments and Environment are the usual ones; InputData is the file or the files to be elaborated (Physical file) and InputTable reports the Logical File related to the Physical one (It is a list of url).

4.4.2.1.1.2 Store&Register task

After the Pilot execution each result must be stored in the GRID-StorageSpace and these result have to be registered somewhere (writing in the OutputTable the URL of the form: **protocol://HostName:port/complete_Path**) hw the choice is left to the application-Manager.



The IN-Data GateWay is the channel used to up-load into the grid files, and it is accessed only during the Preparation phase; the OUT-Data GateWay is the channel used to down-load from the grid the files product. It is accessed only during the Completion phase.

4.5 CASPAR AND GRIDIFICATION

Suppose the first merging attempt is related to the elaboration of GOME data.

Start

```
///All Pilot operation HERE for EO-computation///
```

```
Collect result into RES.tgz
```

```
move result to ESRIN-SE ---->>> *
```

And this could be done with:





```
>>-copy-<<
```

```
URI_1::file://`pwd`/RES.tgz
```

```
URI_2:: [gsiftp] ://theMachine.esa.int/data1/eo/RESULT/MyResult/RES.tgz
```

```
End
```

The CASPAR could pilot the search and discovery of data set requested, in a half-automatized manner [for this first step].

4.5.1 CASPAR & GRID as a Semantic Grid

The architecture could be idealized as composed by a number of archive stores, each of which is fronted by a gateway with a Web/SOAP/Grid interface. The gateway organizes access to meta-data.

The gateway organizes delivery of preserved objects, subject to the satisfaction of access rights.

4.5.1.1 SEMANTIC GRID

The semantic grid is the application of the principles of the semantic Web applied to the grid environment, an extension of the current grid in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation.

In fact, the semantic grid is often viewed as the result of combining grids and semantic Web technology upon a graph of increasing integration and data computation, as shown in the Figure.

There are really two aspects to the semantic grid: the discovery of available resources for processing the data and the ability to integrate the data.

For Data-Grid's (those sharing disk and storage space, instead of providing CPU power), which is how many see the Web already acting, using grid technology (Web services, security, etc.) to provide links and connectivity between information will provide an efficient way of storing information.

For example, with a semantic grid component that stores photos, combined with a semantic grid that stores video material, it should be possible to make connections and associations.

4.5.1.1.1 HOW can we merge CASPAR with Grid's?

CASPAR is a project for the preservation of the Information, an Ontology pioneering preservation environment, the Grid itself is a scheme of protocols to follow for developing a distributed-computing-system.

I'll list the ideas in an order that reflects the difficulty of integration.

The first idea is that CASPAR aids the recovery of Data and the ingestion. It could extend the Replica-Catalog enabling semantic-data-extraction.

It could help the Information-Index during Grid-Services discovery;

So we can check for the existence of a particular resource basing on its meaning;

The Grid-Job itself will be capable of search for some auxiliary data.

Suppose that a scientist needs to configure a complex work-flow of components and to setup the complex scripts required to execute each component like a Grid-job on a specific grid resource: for example the GOME test/execution. One of the tasks involved is a search in certain "area" for a time period, where observations are processed through overlaps and range-recovery. This may involve composing a work-flow of hundreds of jobs and executing them in appropriate computing resources on the grid.

The approach to the CASPAR-GRIDIFICATION problem could be the usage of an automated technique combined with knowledge representation and reasoning in order to specify the





capabilities and requirements of those components as well as the capabilities offered by computing and storage resources (resource here “in the grid sense”) available on the grid.

The system that CASPAR will build-up could use the MetaData-Catalog-Service to perform the mapping between application-specific attributes and logical file names of existing data products.

To orchestrate work-flow we can use GridEngine or DAG-MAN.

What CASPAR will demonstrate is the preservation of the Information; it could also develop ontologies that describe computing and storage resources available on the grid, so that they can be used for matchmaking and dynamic resource allocation.

The goal is to extend existing models with rich definitions of object classes that capture the capabilities of the resources available through the Grid.

For example, to run a Java program requires an ulterior script that assigns a host computer that can run JVM and is running a scheduler (LSF for example), indicates the physical location of the code files, and moves the code to the right places.

CASPAR could be used in this sense, for the resource ontologies and descriptions to select resources automatically given the requirements of each of the application components in a work-flow, and since there is a dynamic context in which the grid-resources live, a static planning and scheduling-technique is a strong constraint. This scenario depicts CASPAR as an add-on (or a meta-service) that could be inserted into GRID.

From the other side we can see a different panorama.

4.5.1.2 CASPAR using the GRID.

If we talk about CASPAR using the terms of web-services, and we have well known what's the meaning of that distribution of services, then we can speak as well as using terms like grid-services.

Substantially a grid-service is an “extended” web-service, as aspect of grid usage made by CASPAR could be the extendibility managing.

Suppose we add/subtract continuously services (i.e. machines representing services) to/from the system. Maybe could be useful to have a service-broker that points to the available servers offering similar services.

Another aspect is the dynamic reconfiguration for services. If we deal with non-static services during time then a grid-based service could be useful.

4.5.2 Completion Procedure &

4.5.2.1.1.1 Publish

The completion procedure basically interacts with the Results publication task.

At the end of publication task, in the CompletionPhase, the subsequent Clean task has to remove all data that are in the Working job directory (Remotely) and the Working job-directory itself.

Locally Grid-Engine deletes all the useless files for the Job, conserving the OutputTable, which indicates the URLs of the result files.

4.5.3 Grid-Engine Job CREATION

4.5.3.1.1.1

IN-BOX consists of 4+1 parts as seen by AdmMgr:

Arguments must be written in a text file called **in.Arguments** that is a complementary part to the pilot file;





StandardInput must be written in a text file called **in.StandardInput** that is another complementary part to the pilot file;

InputTable must be written in a text file called **in.InputTable** that is another complementary part to the pilot file. The InputTable file must list the LogicalNames of requested files.

InputData is the complete set of files that the Computational-Algorithm has to elaborate. They must be readable by the user as indicated by proxy's permissions *¹.

Environment as a list of a name=value pair must be written in a text file called **in.Environment** *².

The 4 files [in.Arguments, in.StandardInput, in.InputTable, in.Environment] must be present in the WorkingDir at the end of the userPrepare program either built as templates directly or maybe not directly, as a result of the userPrepare's procedures subset.

OUT-BOX consists of 3+1 parts as seen by AdmMgr:

StandardOutput of the computational-application must be redirected in a file called **out.StandardOutput** .

StandardError of the computational-application must be redirected in a file called **out.StandardError** .

OutputData is the complete set of files that the Computational-Algorithm produces during its execution. This set must be stored in the Grid (or could be leaved in the temporary-Working-Remote Directory on the Computing Element as decided by the AppMgr).

OutputTable must be written in a text file called **out.OutputTable**. It is a file that lists the LogicalNames of produced files *.

The 3 files [out.StandardOutput, out.StandardError, out.OutputTable] at the end of remote Computational-Process must be leaved in the temporary-Working-Remote Directory on the Computing Element *.

4.6 GRID GLOSSARY

Computational-Algorithm The program furnished by user in the original form, as executed on the userMachine.

Pilot-program [Pilot] Is the computational-algorithm modified in such a way it could be executed remotely in a Grid-Engine context.

IN-BOX A word that points to a concept of the Grid-Engine Job: all the data that are "eated" by Computational-Algorithm.

OUT-BOX A word that points to a concept of the Grid-Engine Job: all the data that are "produced" by Computational-Algorithm.

LogicalFile Is the pointer that could be used by client to retrieve the file requested from Storage-Spaces in (or "near") the GRID.

temporary-Working-Remote Directory Is the directory created by Grid-Engine for the Job.

Computing Element The remote machine used for the execution of the Pilot program.





5 INTEGRATION REPORT FROM IRCAM

In year 2000, IRCAM adopted the MUSTICA system, adopting, on line, the same technique used previously when the documenting material was organized throughout the “Cahier D’Exploitation”.

During the project., the integration between the MUSTICA system and CASPAR has been organized integrating the CASPAR functionalities/components into the preexisting preservation system.

5.1 CASPAR CUSTOMISATION BY IRCAM

The following table report the work performed by the IRCAM staff in order to adapt and customize the CASPAR components

Component	Integration activity report	Feedbacks
DAMS	Integrated very early Definition of a responsibility matrix: ≠ users, ≠ access rights Users : « Musical Assistant » (RIM) Engineer developer Archivist	
POM (Preservation Orchestration)	Specific functions for accessing notifications (demo)	“Pending” object: management of consumed alert and notification – alert status “Grouping”: classification of alert based on element of DCKB. Expertise contains RIModules of DCProfile [GAP – POM interaction expected in rel.0.3 – m31] - topic and expertise “ACK” to producer: option to be checked “ToDo list” as a property of notification
AUTH	Made a local implementation Conforming with original specs	Need to attach different AP to different steps of the lifecycle Need to have templates and customization(curator creates templates, musical assistant customize) ≠ Authenticity protocols for the same object (an audio effect), depending on the work where is used (≠ significant properties)
FIND	Queries made on the basis of RQL	Enhancements needed to





	language	validation tools Solved a problem of Unique Ids (needed in queries)
GapManager	Deployed and tested a local version of the component Defined modules and dependencies	Requirement for clearing the repository in order to include only the UL modules and profiles (done) Provide with a well documented API and examples for the integration of the GapManager with the ICSRiM Archival System (use of the existent API)
CNRS Prototype	Deployed and tested a local version of the component http://kia1.leeds.ac.uk/eleni/proto_v3/prototype.xul Defined new terminologies used by the tool for creating RepInfo objects Modified methods for exporting RepInfo objects in RDF format	Defined new requirements for the next version of the tool Export RDF files using the name of an element and not the elementID Include comments on the RDF export

5.2 OVERVIEW OF INTEGRATED CASPAR FUNCTIONALITIES

Searching the DAMS for user





in-memory filtering by date interval

in-memory filtering by text

Composer(s)	Title	Creation	Ver.	Duration	Category_1	Category_2	Category_3	E	P	S
Tutsoku Hans	DISTANCE LIQUIDE	13/01/2007 0.00.00	v1	0:13:00	TEST	NOT SPECIFIED	NOT SPECIFIED			
Tutsoku Hans	DISTANCE LIQUIDE	13/01/2007 0.00.00	v1	0:13:00	TEST	NOT SPECIFIED	NOT SPECIFIED			
Tutsoku Hans	DISTANCE LIQUIDE	13/01/2007 0.00.00	v1	0:13:00	TEST	NOT SPECIFIED	NOT SPECIFIED			
Tutsoku Hans	DISTANCE LIQUIDE	13/01/2007 0.00.00	v1	0:13:00	TEST	NOT SPECIFIED	NOT SPECIFIED			
Bonardi Alain	early in the morning	03/03/2008 8.05.00	v1	0:24:00	TEST	NOT SPECIFIED	NOT SPECIFIED			
Bonardi Alain	late in the night	05/03/2008 23.10.00	v2	0:24:00	TEST	NOT SPECIFIED	NOT SPECIFIED			
Bathelemy Jerome	new ADDDD	29/02/2008 18.46.00	v1	0:02:00	TEST	NOT SPECIFIED	NOT SPECIFIED			

searching the FIND for all instances

select F1_Work

Composer(s)	Title	Creation	Ver.	Duration	Category_1	Category_2	Category_3	E	S
Tutsoku Hans	DISTANCE LIQUIDE	13/01/2007 0.00.00	v1	0:13:00	TEST	NOT SPECIFIED	NOT SPECIFIED		
Boutard Guillaume Clavarella Raphael	what people expected	27/02/2008 15.04.00	v1	0:20:44	TEST	NOT SPECIFIED	NOT SPECIFIED		

searching the POM for all works containing an item notified

Version: v1 Title: DISTANCE LIQUIDE Comments: TEST BY M. GATT
 Creation date: 13/01/2007 0.00.00 Description: DISTANCE LIQUIDE.xml
 Description: DISTANCE LIQUIDE.xml

Contents

Category 1: DOCUMENT THE WORK Category 2: ASSET DESCRIPTION Category 3: NOT SPECIFIED Actor: GUILLAUME

Title: [] Version: [] Release date: []
 Description: []

adding a new notification to POM for the specified item





in-memory filtering by date_interval

in-memory filtering by text

NOTIFICATIONS list

From: 20/01/2009 To: 20/01/2009 Search

Actor	Title	Description	Release date	Category 1	Category 2	Category 3	E
Bonardi Alain	Reverb	ingest	26/11/2008 13.17.00	NOT SPECIFIED	NOT SPECIFIED	NOT SPECIFIED	
Bonardi Alain	Rev4	ingest	26/11/2008 13.17.00	NOT SPECIFIED	NOT SPECIFIED	NOT SPECIFIED	
Bonardi Alain	test1KHz	ingest	26/11/2008 13.17.00	NOT SPECIFIED	NOT SPECIFIED	NOT SPECIFIED	
Barthelemy Jerome	Ref. sig. 10KHz	Modification in standard for calibrating	19/11/2008 1.08.00	NOT SPECIFIED	NOT SPECIFIED	NOT SPECIFIED	
Boutard Guillaume	Audio AP v1	new release	19/11/2008 10.39.00	NOT SPECIFIED	NOT SPECIFIED	NOT SPECIFIED	
Bonardi Alain	Audio FX v1	new release	19/11/2008 10.39.00	NOT SPECIFIED	NOT SPECIFIED	NOT SPECIFIED	
Lavarella Raffaele	ProTools 8.0	next release	19/11/2008 10.39.00	NOT SPECIFIED	NOT SPECIFIED	NOT SPECIFIED	
Barthelemy Jerome	Ref. sig. 100Hz	obsolete	19/11/2008 2.31.00	NOT SPECIFIED	NOT SPECIFIED	NOT SPECIFIED	

Page 1

searching the POM for all the "opened" notifications

EVENT ITEMS list

Date: 28/02/2008 15:30:00 Location: Cite de la musique Category 1: NOT SPECIFIED Copy

Name: NOT SPECIFIED Description: MUSIC TOMORROW Category 2: NOT SPECIFIED

Duration: 0:00:45 Comment: SUNDAY PERFORMANCE Category 3: NOT SPECIFIED

WORK

Work code: 1 Version: v1 Title: BLOB FROM THE DEEP SPACE Comments: FOR TEST PURPOSE

Creation date: 27/02/2008 15:04:00 Description:

Description	Comments	Work code	Version	Date	Description	Category 1	Category 2	Category 3	RI	FDI	AP	W	NT
artbw	hmjgm	1		01/12/2008	DOCUMENT THE WORK	NEWSPAPER ARTICLE	NOT SPECIFIED	NOT SPECIFIED					
blob - electronics				01/11/2000	PART OF THE WORK	RT LIBRARY	NOT SPECIFIED	NOT SPECIFIED					
ref 100Hz audio sample	for calibration purposes			01/11/2000	PART OF THE WORK	SCORE	NOT SPECIFIED	NOT SPECIFIED					
ref 100Hz audio sample				01/11/2000	PART OF THE WORK	HALL PROGRAM	NOT SPECIFIED	NOT SPECIFIED					

New...

select I from {I}.P106B_forms_part_of {P1}.{P2} P106B_forms_part_of {E1}.{E2} R22B_was_created_by {EC1}.{EC2} R49F_created_a_realisation_of {W}





6 UNESCO INTEGRATION

In UNESCO significant effort to create an application, initially as part of the testbed work, which will help the World Heritage team in their preservation work; this section describes that application.

6.1 EWE: AN EXTENSIBLE TOOL FOR THE PRESERVATION OF VHRP DATA

6.1.1 Vision

This section describes the main strategy followed for the design of the Cultural Testbed Showcase application. Basically, this application aims to give complete support to people involved in instances of virtual-heritage-reproduction (VHRP) offering the use of CASPAR technologies to preserve their data.

EWE Summarizing, EWE (EWE is an Extensible Web tool for the digital preservation of VHRP data), addresses the following issues:

providing an online tool for the definition of vocabularies and schemas,

offering functionalities for the description (using vocabularies already defined) of projects, activities and digital objects used or produced during a generic VHRP, and

a transparency layer that allows users to preserve their data in a OAIS-compliant manner using a subset of the CASPAR Key Component.

According to several studies in the UNESCO domain related to testbed activities that clearly define UNESCO's role as a data collector, there is a strong need for a tool that can deal with an enormous degree of heterogeneity coming from several data sources.

UNESCO collects data encoded in different formats and produced with different processes from different data providers. Each data provider uses its own vocabulary, its own methodology and produces files with different software. Note: All this never comes documented to UNESCO, therefore the need to start providing tools for such a documentation. Just to give a whole overview of the unleashed heterogeneity, consider that the scenario depicted above can be accomplished using several different software and the resulting data can be encoded in a group of different of formats.

EWE tries to address these issues providing functionalities that allow users to define their own vocabulary and to describe their projects using the terms contained within. Once these metadata are provided, EWE translates the schema into an RDFS schema extension of the CIDOC-CRM Ontology and then converts the metadata into instances of such schema. This process aims to produce DescInfo and RepInfo to be packed in a OAIS AIP.

The following document provides a minimum overview of the EWE internal data model, its architecture, a basic use case and an implementation plan.

6.1.2 Data model

The main aim of the *Conceptual Model for the XML Project Description Language Specification* [10] document is to describe the conceptual model behind the XML Project Description Language. This latter is an XML language for the internal representation of projects and activities to be handled with EWE. Following a bottom-up approach all the main entities are defined and an informative example is also shown.

It has been decided to not include here such specifications just for the sake of simplicity. Please refer to the document and presentation provided at the Cultural Testbed Authenticity Meeting [ref] for more details.

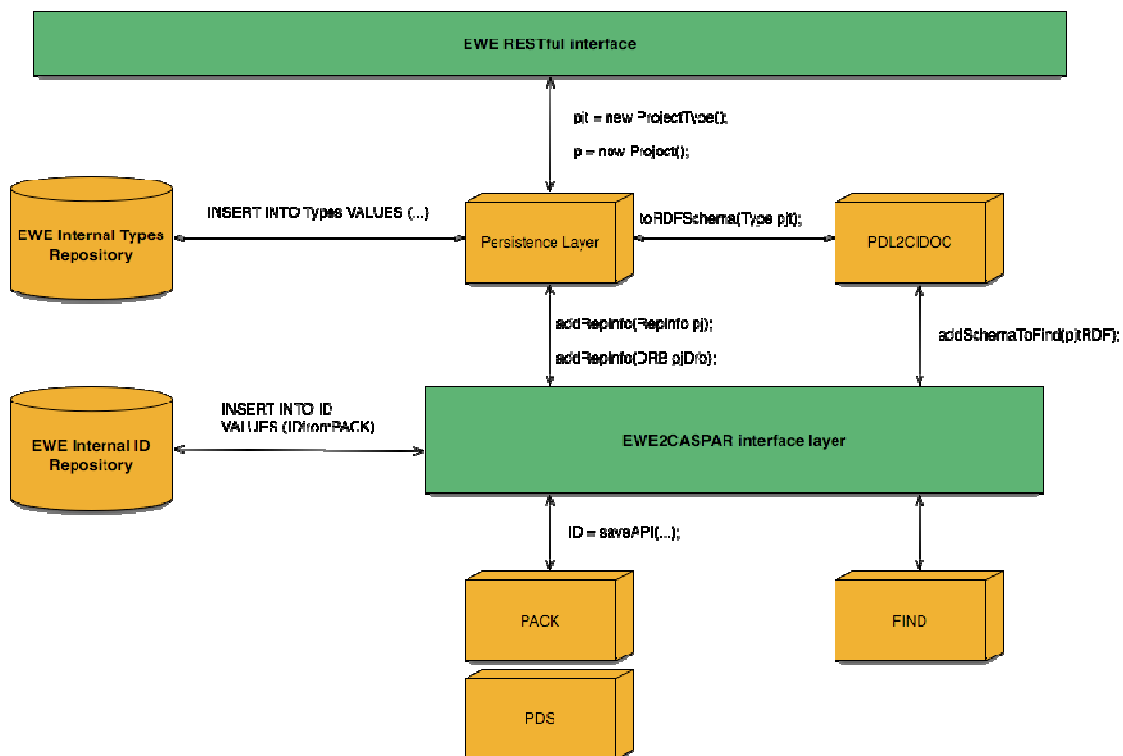




The key concept here is that projects can be described in terms of their activities and their digital objects, and such descriptions (encoded in a XML) translated into an RDFSchema that is an extension of the CIDOC-CRM ontology.

6.1.3 Architecture

The following picture shows the EWE high-level architecture. Each component is therefore described.



The **EWE RESTful interface provides** a set of RESTful APIs allowing users to define project types. Every project type is described by a set of activities each dealing with a type of digital object (i.e. files). Since the final aim of the application is to produce data that can be translated into several RepInfos and DescInfos, the concept of Digital Object Type is the main point here.

In fact, since a Digital Object Type is an XML document describing the format of a specific file with others data related to the production and to context of it, its translation in RDF make possible to store a DescInfo within an AIP in easy way. The feasibility of achieving this using XSPARQL[11] is currently under discussion.

Moreover, a Digital Object Type, embeds all the structural information (more specifically Structural RepInfos) needed for the long-preservation term in a low coupled way: in fact, a separated and specific DRB File is attached to every Digital Object Type.

This component also acts as a transparency layer that hides all the OAIS technologies that underly the application. Furthermore, the choice of exposing the application functionalities as a REST service allows us to build several different Web applications on top achieving a high degree of extensibility.





6.1.4 EWE Core

This component has the application business logic responsibility. Basically, it catch user input, stores and retrieves types and instances in a relational database, converts them into XML to be provided as output of the REST APIs, sends them to the component for the RDFSschema/RDF translation and, finally, is responsible for sending the overall data for CASPAR ingestion/retrieval.

6.1.4.1 PDL2CIDOC

This is the component responsible for converting instances of the two EWE internal languages into a formalism to be sent to the CASPAR Find component as DescInfo. More specifically:

An instance A of the EWE XML Project Description Language was converted into one RDFSschema A_{RDF} that is an extension of the CIDOC-CRM Ontology while,

An instance a of the EWE XML Project Instance Description Language (that represents a concrete project compliant to the description A) was converted into an instance of the A_{RDF} schema.

6.1.4.2 EWE Internal Types/Instances Repository

Mainly, a relational database where types and instances are stored.

6.1.4.3 EWE2CASPAR Interface Layer

This module acts as a proxy between the concrete CASPAR Key Components (more specifically PACK and FIND) and the EWE internal data representation. It also provides storage in a relational database (**EWE Internal ID repository**) for the identifiers obtained from the PACK component in order to fulfil retrieval requirements.





7 CONCLUSIONS

This document has provided an overview of several of the ways in which the CASPAR approach may be integrated into existing systems. It remains to be seen as to whether or not these types of integration will persist. In particular the use of CASPAR components into the developing research e-science infrastructure has yet to be fully investigated, although the roadmap¹ which has been developed by PARSE.Insight does contain the same arguments.

¹ http://www.parse-insight.eu/downloads/PARSE-Insight_D3-5_InterimInsightReport_final.pdf

