FP7-ICT-2007-3-231161

Deliverable 5.2.1
Definition and Design of a PrestoPRIME Reference Architecture for the Integration Framework

W. Allasia (EURIX)

30/06/2010

Document Administrative Table

Document Identifier	PP_WP5_D5.2.1_ArchitectureDesign_R0 Release 0
Filename	PP_WP5_D5.2.1_ArchitectureDesign_R0_v1.02.pdf
Workpackage and Task(s)	WP5 - Integration and validation WP5T2 - Integration of archives, libraries and UGC
Authors (company)	Walter Allasia (EURIX)
Contributors (company)	Matthew Addis (IT Innovation), Werner Bailer (JRS), Guy Ben-Porat (ExLibris), Laurent Boch (RAI), Francesco Gallo (EURIX), Martin Hall-May (IT Innovation), Silvia Lanza (EURIX), Stephen C Phillips (IT Innovation), Elisa Todarello (EURIX), Adil Hasan (ULiv)
Internal Reviewers (company)	Matthew Addis (IT Innovation), Werner Bailer (JRS), Laurent Boch (RAI)
Date	M18 - 30/06/2010
Status	Release
Version	1.02
Type	Deliverable
Deliverable Nature	Report
Dissemination Level	Public
Planned Deliv. Date	M18 - 30/06/2010
This IsPartOf	
This HasPart	
Abstract	Architecture Design of the Integration Framework

DOCUMENT HISTORY

Version	Date	Reason of change	Status	Distribution
v0.01	2010-01-22	Defined index, included wiki contributions	Outline	Confidential
v0.02	2010-01-25	Completed a few sections, fixed typos	Outline	Confidential
v0.03	2010-03-30	Updated State of the Art section	Working Draft	Confidential
v0.04	2010-04-22	Updated SIP and RA sections	Working Draft	Confidential
v0.05	2010-05-05	Updated SIP section	Working Draft	Confidential
v0.06	2010-05-21	Updated SIP and RA sections	Working Draft	Confidential
v0.07	2010-06-14	First Final Draft version	Final Draft	Confidential
v0.08	2010-06-15	Fixed Typos in Final Draft version	Final Draft	Confidential
v1.00	2010-06-30	Release corrected according to feedbacks from internal reviewers	Release	Confidential
v1.01	2010-07-05	Release corrected according to review by UIBK	Release	Confidential
v1.02	2010-07-12	Fonts and page layout modified	Release	Public

Executive Summary

This deliverable provides the technical analysis design of the software framework of PrestoPRIME Preservation Platform, for preserving digital contents of archives and libraries. This document should be considered neither exhaustive nor frozen. It should be rather consisedered an attempt to design a software preservation platform that fulfills the requirements identified within Task 5.1 (D5.1.1 [1]) and which is able to take into account the outcomes provided by Work Packages 2,3,4. A table summarizing the scenarios and how they are covered by the proposed architecture, is available in Section *Conclusions* 6. Deliverable D5.2.1 has to be considered as the *abstract*[1] design of the PrestoPRIME Preservation Platform, the reference architecture and model of the preservation system. There will be two implementations:

- an open source software framework which will be the Deliverable 5.2.2 [3] (prototype). This implementation will be the reference software for the partners and will be useful for experimenting with the other Work Packages' results;
- a commercial system delivered by ExLibris, that, extending Rosetta [4] and implementing the reference architecture, will represent the commercial solution provided by PrestoPRIME ready to be used at the end of the project. This implementation is formalised in Deliverables 5.3 [5, 6] (prototype).

The main reason for having two implementations (instead of one only) is that the first one will be released freely and may be used for experimenting the software outcomes of PrestoPRIME as well as extend the source code in order to implement new features. On one hand it will implement all the specifications defined this document but on the other hand it should be considered as a pure prototype, not engineered and with more attention to the macro functionalities rather than to the performances. From the beginning PrestoPRIME wanted also to provide a professional solution available on the shelf, hence a second implementation has been planned involving one of the market leaders in the field of digital preservation systems such as ExLibris [7], in order to prove the strength of the designed solution and the project results also in a commercial environment. At the end of the third year (Autumn 2011), the two implementations will be available for the test session and test results will be gathered and published in the deliverables Work Package 5 Task 4.

[1]The term *abstract* has to be considered with the meaning defined in UML [2]

Table of contents

1 Introduction

This document represents the reference software definition of the PrestoPRIME Preservation Platform (named P4 system). This document is not exhaustive and completed in the sense that the analysis and design performed point out the main features, leaving the details and further component descriptions to the implementation phase.

The analysis undertaken during the first year has defined the guidelines to follow in our architecture design: first of all the OAIS [8] Open Archival Information System standard, provided by the Consultative Committee for Space Data System, subsequently, the TRAC [9] and related documents adopted by the most important bodies in the field of digital preservation such as National Libraries (LOC, etc.), broadcasters (the project partners and others) and international projects oriented to the realisation of a digital preservation solution.

The reader should keep in mind that this document is a reference software specification and technical languages as well as design practices are commonly used. In order to give an overview, the document is made up of:

- a State of the Art section for architecture, outlinign designs of similar projects;
- a State of the Art section for Submission Information Package (SIP [8]), which included a general discussion on Information Packages and provides some SIP examples from related projects and institutions;
- a section for the specification of PrestoPRIME SIP;
- the architecture analysis and design, which is the core part of the document.

The State of the Art section takes into account a few relevant projects such as CASPAR and SHAMAN, pointing out their results and approaches. It is not a complete SOTA on digital preservation, which is far beyond the scope of this document.

The Information Package sections have been introduced in order to define what is the structure of the contents to be preserved. If we consider the preservation platform as a *black box*, it works with contents coming in and out. The structure of what goes out can be decided later and also can be translated/converted into different ones. The input documents are more critical: the black box platform should know in advance the object to be handled.

This is the reason why we have introduced this section on SIP and also why we spent several months discussing with the other partners trying to find out the best agreement on it. The knowledge of the SIP structure is mandatory for the platform. We decided to place the SIP section just before the Architecture design. It describes the agreed content and metadata structure of the digital items coming into the PrestoPRIME Preservation Platform.

Even if a preservation system is responsible to preserve Information Packages, more specifically AIPs (Archival IPs), their structure is specific of a software implementation.

The SIP is usually a part of an AIP and also the DIP (Dissemination IP) is something smaller than the original AIP. The analysis of the other Work Packages as well as the software implementation descriptions (D5.2.2 [3], D5.3 [5, 6]) will provide the definition of the AIP within PrestoPRIME.

In this document we focus our attention on the software components realising the digital preservation system implementation. A specific format description of the internal Information Packages is not mandatory for the design. For the structure of SIP has been chosen general enough to be able to take into account further extensions and new formats and at the same time detailed enough to be able to manage the ingestion phase, a really simplified solution can adopt this structure for the SIP as well as the AIP and DIP.

Summing up, the following document takes into account: the Scenarios identified in D5.1, the best practices brought in by the partners, the OAIS reference model, the outcomes and analyses of Work Packages 2,3,4. The reader will get a quick overview of the available results from other EU funded projects, then the SIP structure is discussed and finally the architecture analysis and design is presented.

2 State of the Art: Architecture

In order to design the PrestoPRIME integration framework, a comparative analysis of existing implementations of the OAIS Reference Model (summarised in Fig 1) has been carried out. The considered projects are:

- CASPAR [10], Planets [11] (both EU funded under the 6th Framework Program), SHAMAN [12], Keep [13] (both EU funded under the 7th Framework Program)
- Preserv2 [14], KeepIt [15] (funded by JISC [16])
- DSpace [17], Fedora Commons [18] (US)
- Rosetta [4] (Commercial - ExLibris)

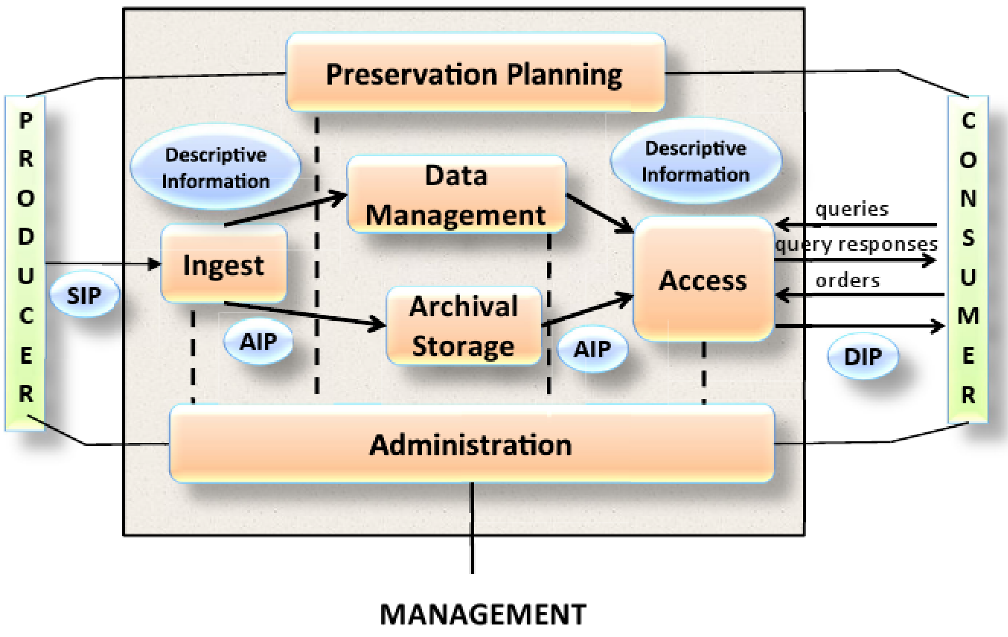

Figure 1: *OAIS Functional Entities [19]*

2.1 CASPAR

Particularly relevant is the CASPAR system [10], which maps the OAIS functional entities to software components. Figure 2 shows the system architecture. This analysis is based on CASPAR Public Deliverables [21],[22], and [23].

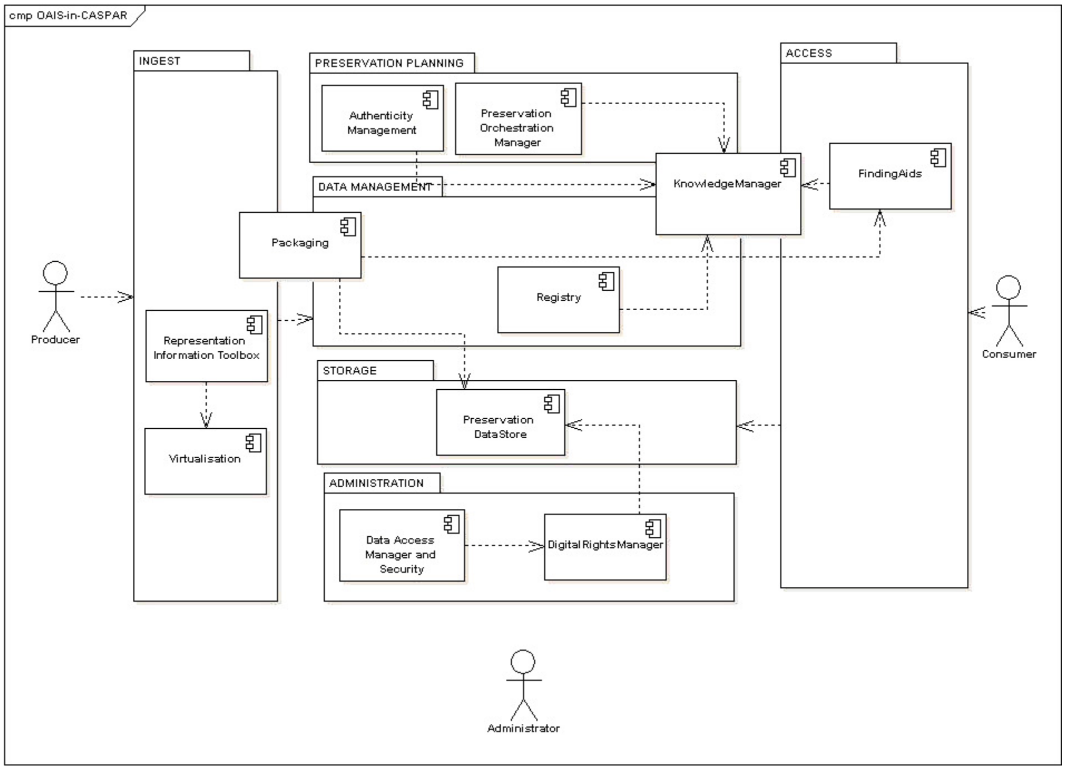

Figure 2: *Mapping OAIS Functional Model to CASPAR Key Components. From [20]*

Ingest

The Ingest functionalities are realized by the following CASPAR components:

- *Representation Information Toolbox and Virtualisation Toolbox* (REPINF)
  The Virtualisation Toolbox is part of the RepInfo Toolbox. This component has the responsibility to hide technical detail through encapsulation.

- *Packaging* (PACK)
  This component's responsibilities are the construction, access, manipulation, validation and delivery of Information Packages (SIP, AIP, DIP).

Preservation Planning

The Preservation Planning functionalities are realized by the following CASPAR components:

- *Preservation Orchestration Manager* (POM)
  This component is an implementation of the Publish-Subscribe pattern. POM receives event notifications from a Data Preserver (with publisher role) for a specific

"topic". A Data Holder (with subscriber role) is registered to the POM in order to receive alerts.

- *Authenticity Manager* (AUTH)
  This component manages the process (Protocol) of authentication of the digital objects, checking identity, authorship and integrity.
- *Knowledge Manager* (KM)
  This component comprises two layers:
  a. SWKM (Semantic Web Knowledge Middleware): a low level manager which provides core services for managing Semantic Web Data. These service are: query, update, import, export.
  b. CKM (Caspar Knowledge Manager): an uppper layer which provides OAIS-like functionalities using the services offered by SWKM.

Data Management

The Data Management functionalities are realized by the following CASPAR components:

- *Registry* (REG)
  The Registry component provides functionalities for storage and retrieval of Representation Information (including Preservation Description Information (PDI)) in a centralised Registry/Repository. It is also responsible for the long-term mantainance of the Representation Information.
- *Packaging* (PACK)
- *Knowledge Manager* (KM)

Archival Storage

The Archival Storage functionalities are realized by the following CASPAR components:

- *Preservation DataStore* (PDS)
  This component is responsible for storing, retrieving, transforming and migrating AIPs. It is based on a three-layered architecture:
  1. a preservation engine layer, which provides an external interface to PDS and implements preservation functions;
  2. an XAM [24] layer, which serves as the storage mid-layer;
  3. an Object Store Device (OSD) [25] layer.

This module was developed by IBM and is not openly available.

Administration

The Administration functionalities are realized by the following CASPAR components:

- *Data Access Manager and Security* (DAMS)
  This component allows managing users, groups and profiles, and access policies.
- *Digital Rights Manager* (DRM)
  This component registers a digital item's creation history and derives the existing Intellectual Property Rights from the creation history. It also allows exporting rights information in terms of instances of the CASPAR Digital Rights Ontology.

Access

The Access functionalities are realized by the following CASPAR components:

- *Finding Aids* (FIND)
  This component exposes interfaces for discovering AIPs through their associated Descriptive Information and for storing and updating this DI, and provides text-query functionalities over DI objects.

Conclusions

CASPAR provides a sofware implementation of the OAIS functional model, adding some enhancements, such as a registry for storing Representation Information, the use of an ontology for representing digital rights. The project is also focused on modelling and managing the Knowledge Base of the Designated Community, an important aspect of OAIS Reference Model.

2.2 Planets: Interoperability Framework (IF)

The Planets[11] project has delivered an OAIS compliant system, the Planets Interoperability Framenwork, which integrates a number of preservation tools. As reported in [26], the Planets Interoperability Framework provides a service-based infrastructure. The core of the IF implementation is based on the Java Enterprise Edition (Java EE 5). The IF installation package includes a pre-configured JBoss application server. This application server provides the container for web-based preservation applications: the Planets Testbed application [27] and the Planets Preservation Planning Tool, PLATO [28].

Figure 3 provides an overview of the interactions that the Planets IF has with ingest, repository, and delivery services, as well as with third-party tools and service providers.

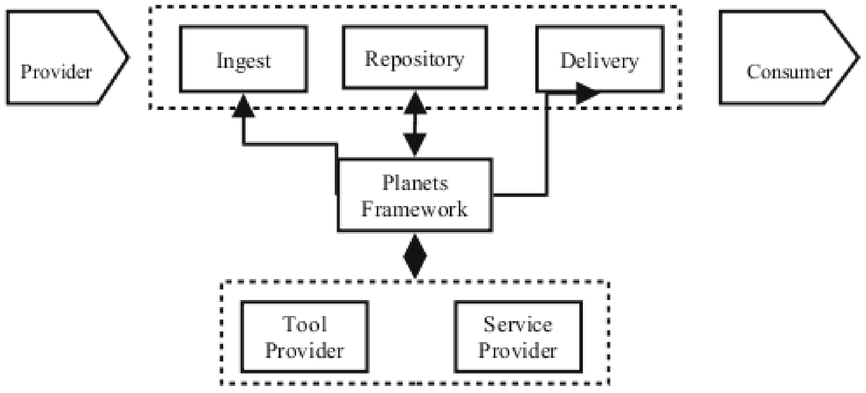

Figure 3: *The Planets software framework interactions. From [26]*

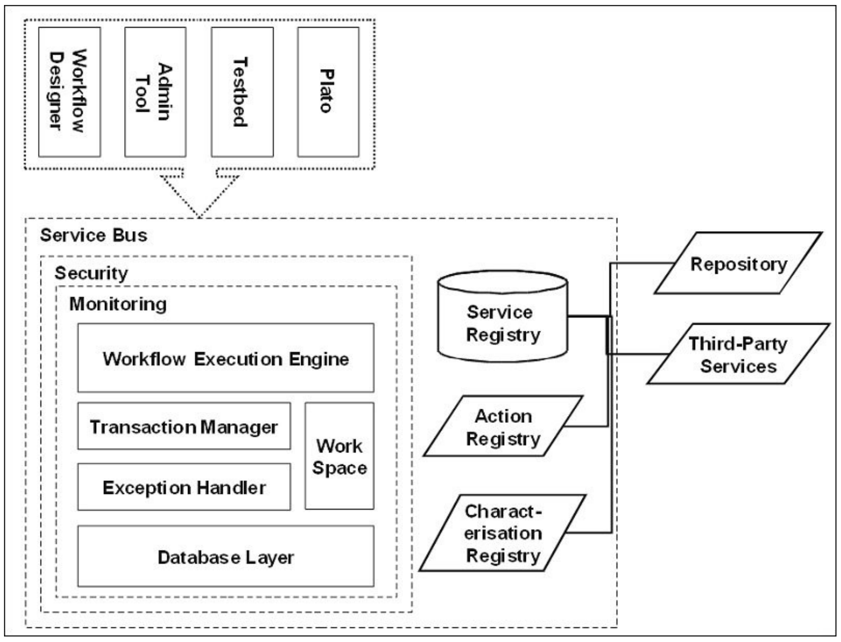

Figure 4: *The Planets Interoperability Framework. From [26]*

Figure 4 shows a more detailed view of the components of the Planets IF and their relationships with Planets applications, repositories and third-party services. The Interoperability Framework establishes a service bus as well as essential shared services and components. These include:

- the security component, which provides authentication and authorization services;
- the monitoring component, which provides flexible monitoring and logging services;
- the workflow execution engine, which takes workflows specified in the Business Process Execution Language (BPEL) and executes them in the context of the available Planets services;
- the transaction manager, which provides roll-back and compensations for complex transactions which may be implemented by workflow elements;
- the exception handler, which provides a uniform set of services to register and handle exceptions that arise during service execution;
- a database or persistence layer;
- workspace services so that workflows have appropriate levels of isolation.

Services have basic definitions registered in the service registry. There are two important specialised registries: one for preservation actions, and one for content characterisation services. In addition, repository services and third-party services are also defined in the service registry.

Ingest

File format recognition is performed upon submission. Known file formats are stored in a so called Representation Information Network (RIN). A Representation Network provides the information required to understand and make use of a particular digital object. This might include information about how to render a digital object as well as information explaining how to understand, interpret or re-use the digital object.

If a matching entry is not found in the RIN for the format of the file to be ingested, a request is passed to the Preservation Watch functionality (see below), which may seek information on this new file format. An update is then made to the RIN. Finally, the metadata describing the characterised object is passed back to the Ingest function.

Preservation Planning

Planets Preservation Planning tool is called PLATO [28]. Four preservation functional entities have been identified which extend the OAIS definition of Preservation Planning.

1. *Preservation Watch Function*
   Preservation Watch monitors or surveys a number of internal and external entities. It

monitors digital objects, metadata, users, the preserving organisation and the wider technological environment in order to provide preservation requirements, update preservation metadata and provide alerts as to when preservation action needs to be taken.

2. *Preservation Planning Function*
   Preservation Planning evaluates requirements and selects the most appropriate preservation solution available. It requests development of new preservation actions when needed, and updates metadata describing new preservation actions.

3. *Preservation Action Function*
   Preservation Action performs actions on digital objects to ensure their continued accessibility. It also develops new preservation actions (for example, migration tools, emulators).

4. *Preservation Characterisation Function*
   Preservation Characterisation provides support to ingest activities and preservation action activities. When digital objects are ingested into a repository, Preservation Characterisation identifies file formats and extracts metadata. When a Preservation Action is performed, Preservation Characterisation characterises the digital objects before and afterwards to enable evaluation of the action.

Archival Storage and Data Management

The Data Registry module of the IF provides storage and access to files and metadata. The Data Registry uses the Apache Jackrabbit implementation of the standard Java Content Repository API. The module exposes a low-level graphical interface which allows users to perform operations on stored content. These operations functionally correspond to OAIS entities Data Management, Access, and Storage. Operations which correspond to OAIS Data Management functionality are: add content file references and associated metadata, save metadata generated by workflow tasks. Operations which correspond to OAIS Access functionality are: browse content/data, search content (by both XPath and SQL queries). Operations which correspond to OAIS Storage functionality are: create, delete, and modify content (e.g. by adding new content, modifying properties, etc.).

Administration

As reported in [29], Planets IF comprises an administration interface (AdminUI), whose main responsibilities are user management and role assignment. It is a web application that makes use of the Acegi Security System [30] for the purposes of authorization and authentication. User information (address, email, password etc.) is stored in a RDBMS. Three types of security (protocol, path and method based security) are implemented. Users may apply for an account. The administrator assigns roles (service provider, for example) to users after their successful registration and these roles will then determine the pages and functions that a user can access and execute.

Access

See Archival Storage and Data Management.

Conclusions

It is possible to find out more about the behaviour of tools and digital objects in a digital preservation setting with Testbed v1.1 [27] (Public Beta), and use version 2.1 of the web-based Planets preservation planning tool (PLATO [28]) .

2.3 SHAMAN

The SHAMAN project [12] develops a framework for the long-term digital preservation framework exploring the Multivalent technology.

From the point of view of the life-cycle of digital objects and of the preservation system, the representation of the SHAMAN Digital Preservation Framework shown in Figure 5 has been proposed. This is a value-chain representaion in which Production, Consumption and Archiving are integrated.

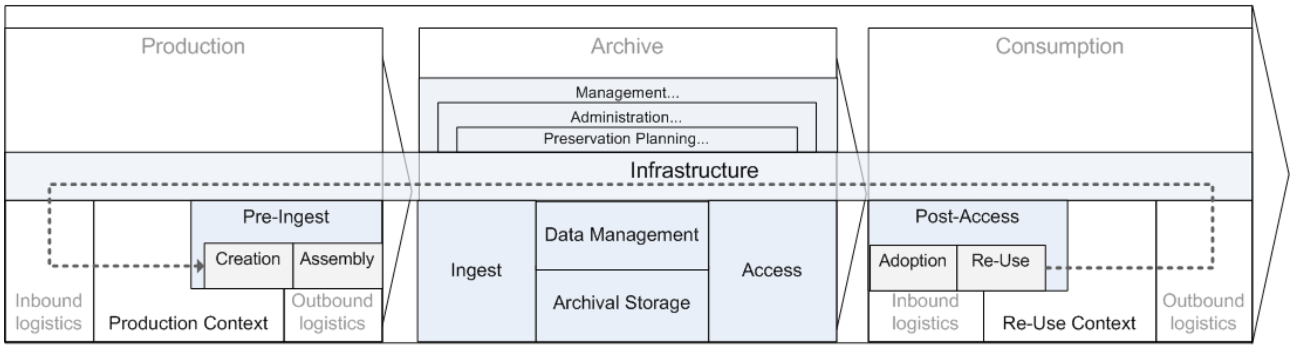

Figure 5: *The scope of the SHAMAN Digital Preservation Framework. From [31]*

Pre-Ingest

Pre-Ingest compirses the production of the digital object, the assembly and validation of the SIP. SIP assembly is not part of the OAIS Reference Model.

Ingest

Ingestion module is responsible for assembling AIPs from SIPs, adding to the package the information needed for long-term preservation (such as policies work-flows records, association of the correct access tool), and validating AIPs. The access tool itself needs to be preserved and is contained within its own AIP.

Preservation Planning

Preservation of content is accomplished with the design of SIP, AIP and DIP, based on abstract descriptions of the life-cycle of the digital object within a specific domain. For example, the SIP includes an abstract description of the process which defines how documents have to be analysed to ensure that they are discoverable. An abstract description of the preservation processes forms part of the AIP. Policies are used to create and manage the SIP, DIP and AIP.

Archival Storage and Data Management

Data Grids are used in SHAMAN as a storage technology. In particular, SHAMAN makes use of iRODS [32] (Integrated Rule-Oriented Data System), the open source successor of SRB[33]. Validation functions are included within the Archival Storage to check that the storage obeys the service level agreements.

Administration

Administration is responsible for demonstrating that the repository is trustworthy. This can be achieved by recording the history of the digital object whilst in the system; ensuring that only authorised access to the data is allowed and recorded; that the integrity of the data is demonstrated and recorded throughout its lifetime in the system; that the system provides non-repudiation (e.g. to ensure acknowledgement of data received from the Producer, or data sent to the Consumer). Administration also manages Archive policies, which can be classified into three main categories: Submission Policies, Preservation Policies and Dissemination Policies.

Access

Data access includes the assembly of the DIP which is made up of the correct access tool along with the data and related metadata. The DIP must be validated to ensure it correctly renders the data. The access tool is packaged up into the DIP along with its preservation environment.

Post-Access

The Post-Access comprises transformation of the object requiring functions that are not part of the preservation system and re-using the object to create a new object that would need to be re-ingested into the archive. These functions lie outside of the OAIS definition.

Conclusions

At the time of the release of this deliverable (June 2010), the SHAMAN project has been busy performing state-of-the-art analyses (such as reviewing OAIS), better understanding the real usage scenarios, defining solution architectures, and developing the first set of demonstrators. In first half of the project, the focus was on the analysis and development of the scenarios.

2.4 Keep

Keeping Emulation Environments Portable (KEEP) [13] is a medium scale research project started on 1 February 2009 co-financed by the 7th Framework Programmes ICT-3-4.3 Digital libraries and technology-enhanced learning priority. The KEEP project will develop an Emulation Access Platform to enable rendering of both static and dynamic digital objects: text, sound, and image files; multimedia documents, websites, databases, videogames etc. [34]

As shown in 6, KEEP is based on the OAISs information model. It will design a preservation metadata application profile from existing metadata specifications such as Dublin Core, PREMIS, MIX and METS. [34]

Identification of the archived digital objects is done using the Planets Characterisation Registry and results in additional metadata about the type of object and its dependencies on hardware and software. Identification of the available preservation tools to render the digital object is also covered by Planets by introducing the registry tools for preservation action. Emulators supported by KEEP will be registered to this registry and can be selected for emulation [35].

Conclusions

As written in [13], KEEP is a framework that contains many emulators. The central goal of KEEP is to provide new tools for accessing any digital object both at present and in the long term. Another objective of the KEEP Project is to determine the suitability of existing data formats for a variety of data transfer tasks.

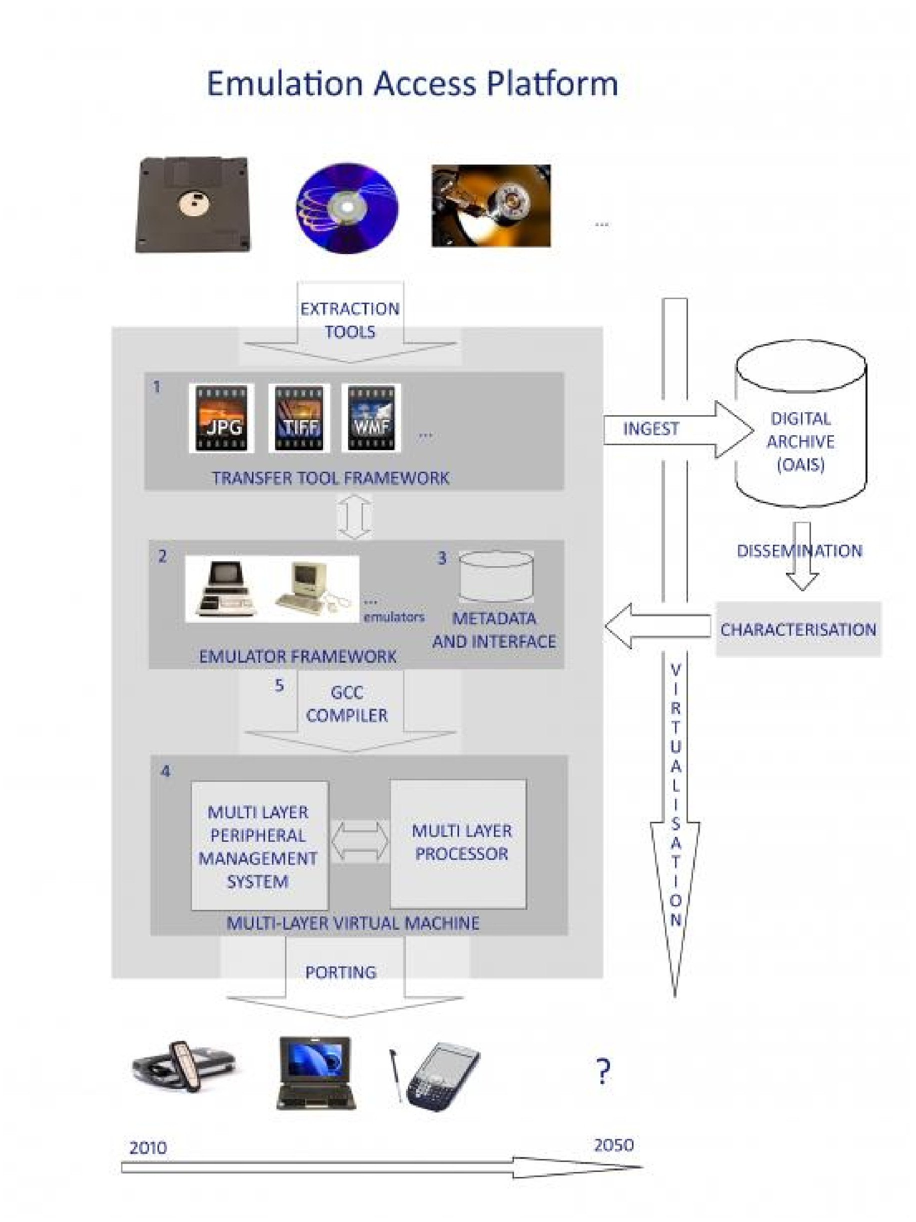

Figure 6: *KEEP Emulation Access Platform. From [13]*

As reported in [35], the results of other European projects such as PrestoSpace or CASPAR as well as the International Internet Preservation Consortium (IIPC) are taken into account by KEEP. More particularly, KEEP has a strong relation to the outcomes of European projects Planets and SHAMAN.

2.5 Preserv 2

Preserv 2 [14] project (from July 2007 to March 2009) was a JISC [16] funded project with the aim of developing digital preservation services for repositories. Preserv 2 is the continuation of the Preserv project, and it is now being mantained under the JISC funded KeepIt project [15] (April 2009 - September 2010). The preservation features developed by Preserv 2 have been included in EPrints [36] repository software since version 3.

Preserv 2 was mainly focused on two of the OAIS functional entities: Archival Storage and Preservation Planning.

Archival Storage

Preserv 2 developed a repository storage controller, a feature included in EPrints 3.2, which enables the repository manager to use multiple storage platforms including local, institutional and cloud based, such as Amazon S3 & Cloudfront and Sun Honeycomb.

Preserv 2 also developed a strategy for the migration of data between two repository softwares with quite distinct data models, from an EPrints repository to a Fedora repository. This was accomplished using OAI-ORE [37].

Preservation Planning

Another large area of work in Preserv 2 was on file format management and an “active preservation” approach. This involves identifying file formats, assessing the risks posed by those formats and taking action to obviate the risks where that could be justified. This led to a new approach combining file format management with automated storage management, called “smart storage”. “Smart storage combines an underlying passive storage approach with the intelligence provided through services.” The Preserv 2 implementation of smart storage is shown in Figure 7.

In Preserv 1, file format identification was accomplished through the development of PRONOM-ROAR [38], [39], a web service presenting format profiles of more than 200 EPrints and DSpace repositories; while in Preserv 2 file format classification was performed using the Preserv2 EPrints preservation toolkit, which wraps DROID [40] (see Figure 7) in an EPrints repository environment. Additional tools are included: a calendar-based scheduler, a harvester to retrieve the latest stored content from a repository, messaging services to record the history of events. These tools are designed to work with the core format identification tool DROID. According to the tests, the DROID implementation seems to give better results than the PRONOM-ROAR one.

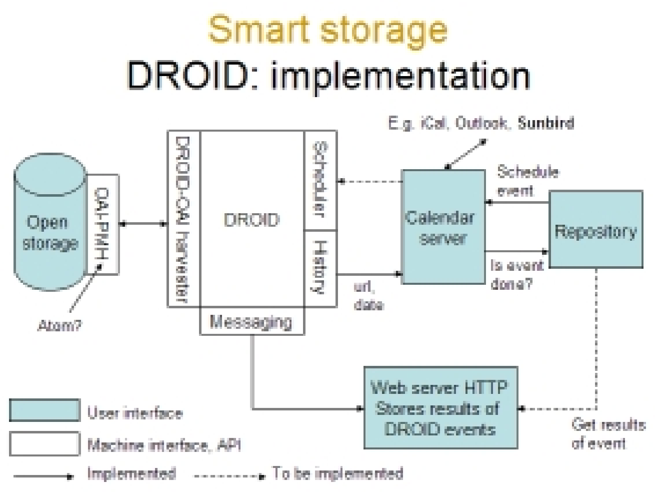

Figure 7: *Preserv 2 schematic, Sept. 2008. From [14]*

Among the services included in Preserv 2 EPrints preservation toolkit, there's a functionality for adding preservation metadata to the digital objects during ingestion. These metadata are a subset of the PREMIS [41] Data Dictionary.

Conclusions

Preserv 2 project ended in 2009 and is currently taken forward in project KeepIt. The reseach activity carried out in Preserv 2 resulted in the inclusion of the Storage Controller in EPrints v3.2. The storage controller enables the repository manager to use multiple storage platforms. Preserv 2 also tested file format identification tools, i.e. PRONOM-ROAR and DROID and it began the implementation of the so called "smart storage", which consists in adding format-specific preservation services to customary passive storage.

2.6 DSpace

DSpace [17] system architecture is shown in Figure 8. The system comprises three layers:

1. The application layer is responsible for interactions of the system with the world outside of the individual DSpace installation.
2. The business logic layer manages the system internally.

3. The storage layer is responsible for the physical storage of metadata and content.

The DSpace data model is described in [42]. It consists of a structure of classes in which the basic archival element is called `Item`.

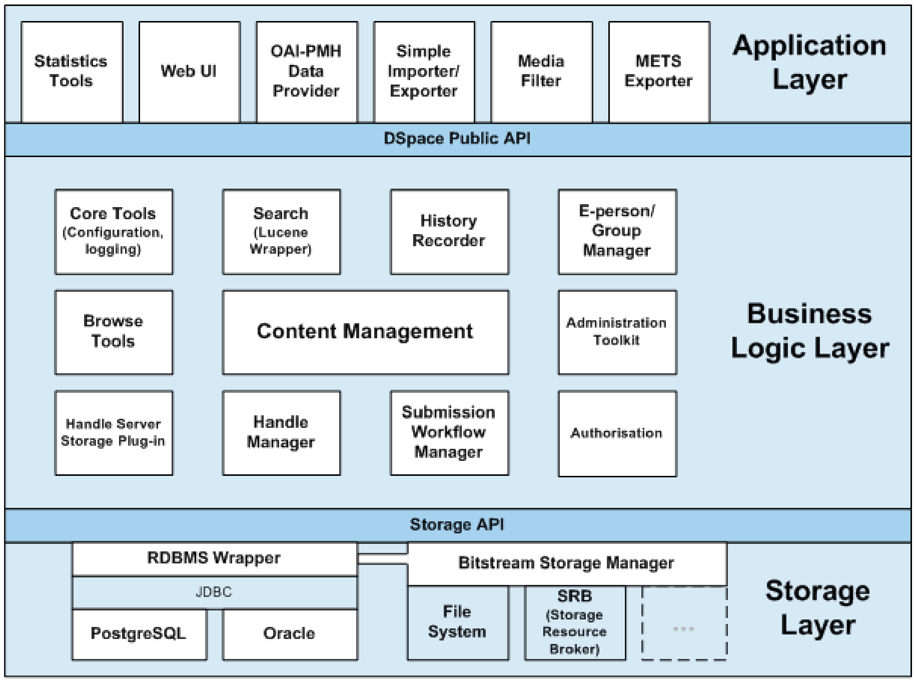

Figure 8: *DSpace System Architecture. From [42]*

Ingest

DSpace uses a workflow manager to model submission steps. The system implements its own workflow manager, which models the workflows as 5 steps: SUBMIT, three intermediate steps (STEP1, STEP2, STEP3), and ARCHIVE.

Preservation Planning

Ingested bitstreams have a format associated to them, which is referred in a unique and consistent way. Each format has a support level associated with it, this level can be one of: Supported (the system is likely to have appropriate preservation services for this format); Known (the format is recognised, but there are no preservation services for it, the bitstream is preserved as-is); Unsupported (the format is not recognised). The bitstream formats recognised by the system and levels of support are similarly stored in the bitstream format registry. What formats to put in the registry is up to the user at installation time or later through the AdministrationUI. Moreover, the data model comprises preservation metadata, such as provenance and fixity (checksum).

Data Management

This module provides methods for manipulating content stored in the system's database, such creating `Item`s, collections of `Item`s, editing metadata.

Archival Storage

DSpace stores information about users, content metadata, content organisation and state of currently-running workflows in a relational database. For storing content, the system provides methods for storing the server file system or using a SRB [33].

Administration

Packages which implement this functionality provide common functionalities such as managing system configuration, creation and manipulate EPerson (i.e. users registered in the system), management of policies, (which allow an EPerson to perform actions explicitly listed in the policy on the defined resources).

DSpace solves the problem of creating long-lasting identifiers for stored `Items` reliyng on an external external service: CNRI Handle System [43]. DSpace uses Handles as a means of assigning globally unique identifiers to objects. Each site running DSpace needs to obtain a Handle 'prefix' from CNRI.

Access

Classes which implement search functionality are basically a wrapper around Apache Lucene [44] text search engine. These classes have methods for adding `Item`s to the index, updating the index and rebuilding the index from scratch. Fields to be indexed can be configured. Methods for browsing `Item`s stored in the system are available as well.

Conclusions

DSpace is a system supported by a large community of users (more than 700 organizations are currently using it) and developers. Latest stable release is 1.6, released on 2 March 2010. A new enhanced version, DSpace 2.0, is currently under developement.

2.7 Fedora Commons

The Fedora Repository is a product of Fedora Commons [18]. It is able to store digital content items and related information in any format. [45]

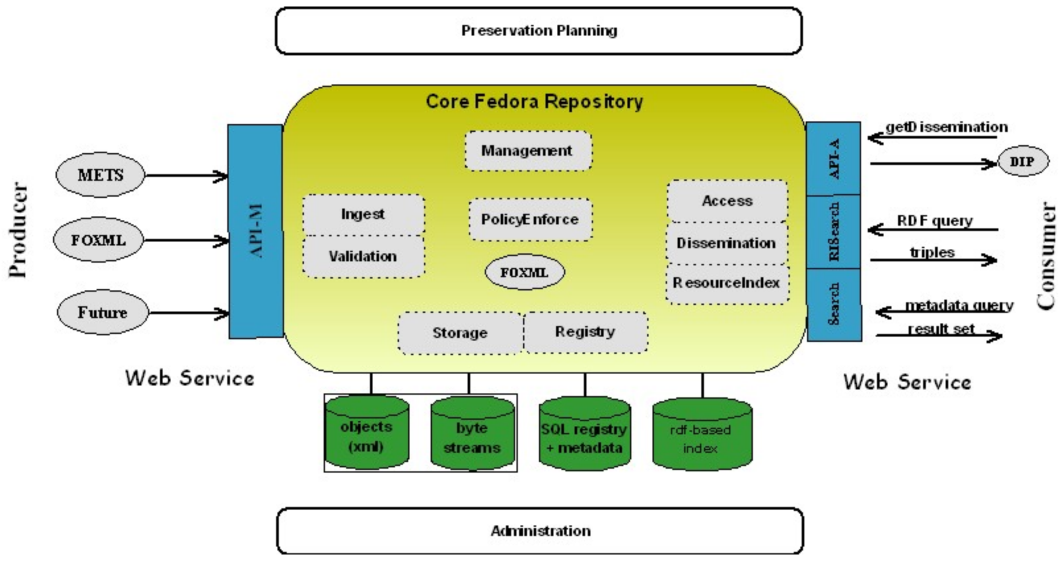

Figure 9: *Fedora framework. From [46]*

Ingest

The Directory Ingest Service is a service to ingest a hierarchical directory of files into a Fedora repository. The service will accept a Submission Information Package (SIP) in the form of a ZIP archive that contains directories of files along with a METS-based manifest file that describes the directory hierarchy. The default parent-child relationships that characterise a directory hierarchy can be overridden and refined to have other semantic meaning (e.g., collection-member, folder-document). Upon receipt of the SIP, the Directory Ingest service will process the ZIP archive and create a Fedora digital object in the repository for every file and every directory, plus it will record the relationships among them in Fedora's RDF-based relationships datastream. A web-based client for creating the SIP is available [47].

Archival Storage and Preservation Planning

Fedoras archival and preservation capabilities include:

- XML: Fedora objects XML and the schema upon which they are based are preserved at ingest, during storage, and at export.

- Content Versioning: Fedora repositories offer implementers the option of versioning data objects. When a data object is versioned, the objects audit trail is updated to reflect the changes made to the object, when the change was made and by whom and a new version of the modified data is added to the objects XML. This new Datastream cascades from the original and is numbered to show the relationship between original and version. This allows users to retrieve older versions of a data object by performing a date/time search and retrieval, or the most current version if the date/time criteria are not included in the search.
- Object to Object Relationships: relationships between objects can be stored via the metadata included in the objects. This allows implementers to link together related objects into parent/child relationships.
- Event History: every object in a Fedora repository contains an audit trail, which preserves a record of every change made to the object.

Data Management and Administration

One of the goals for Fedora's development is the Integrated Management: efficient management by repository administrators not only of the data and metadata in a repository, but also of the supporting programs, services and tools that make presentation of that data and metadata possible. [45]

Access

The model digital object contains Datastreams [45] and a set of object properties . A set of access points is defined for each object using the methods described below. Each access point is capable of disseminating a “representation” of the digital object. A representation may be considered a defined expression of the essential characteristics of the content. Each access point is identified by a URI that conforms to the Fedora ”info” URI scheme.

Fedora provides several protocol-based APIs to access digital objects. These protocols can be used both to access the representation and to obtain associated metadata at the same access point. By default, Fedora creates one access point for each Datastream to use for direct dissemination of its content. [46]

Conclusions

The Fedora development team has released an initial set of services (Directory Ingest and OAI Provider), and continues to develop new services during Fedora Phase 2 (2005-2007) and beyond, especially services for workflow and preservation. [46]

It is possibile to use and test Fedora 3.0 release, which introduces the Content Model Architecture. The Content Model Architecture (CMA) has added a new system-recognised Fedora Object type, the Content Model Object. Among the improvements this architectural change provides, there's the fact that operations or behaviors are now attached to objects at the content model level. The traditional ”disseminator” has been replaced by the CMA functionality.[48]

2.8 Rosetta

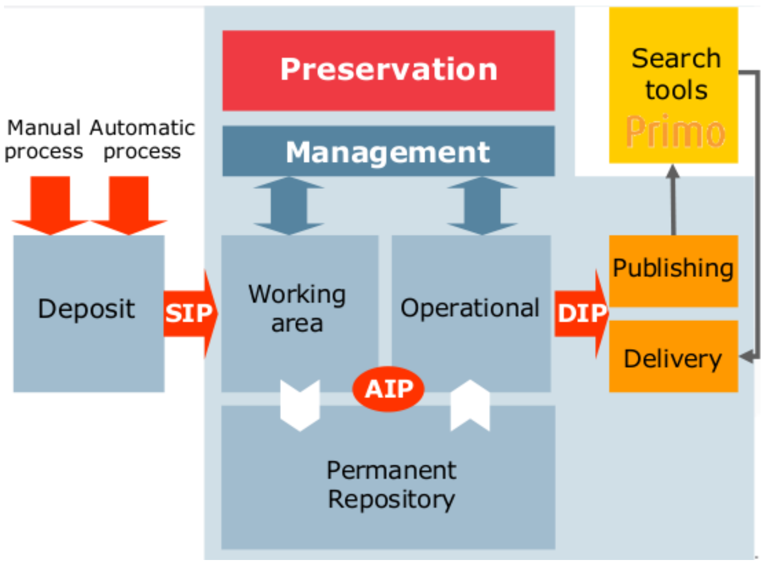

Figure 10: *Rosetta architecture. From [49]*

The OAIS model is used in the Rosetta system across its modules. The correspondence between OAIS fuinctional entities and Rosettsa software components is shown in Figure 11.

For further details on Rosetta architecture refer to PrestoPRIME Deliverable 3.1.1 [50].

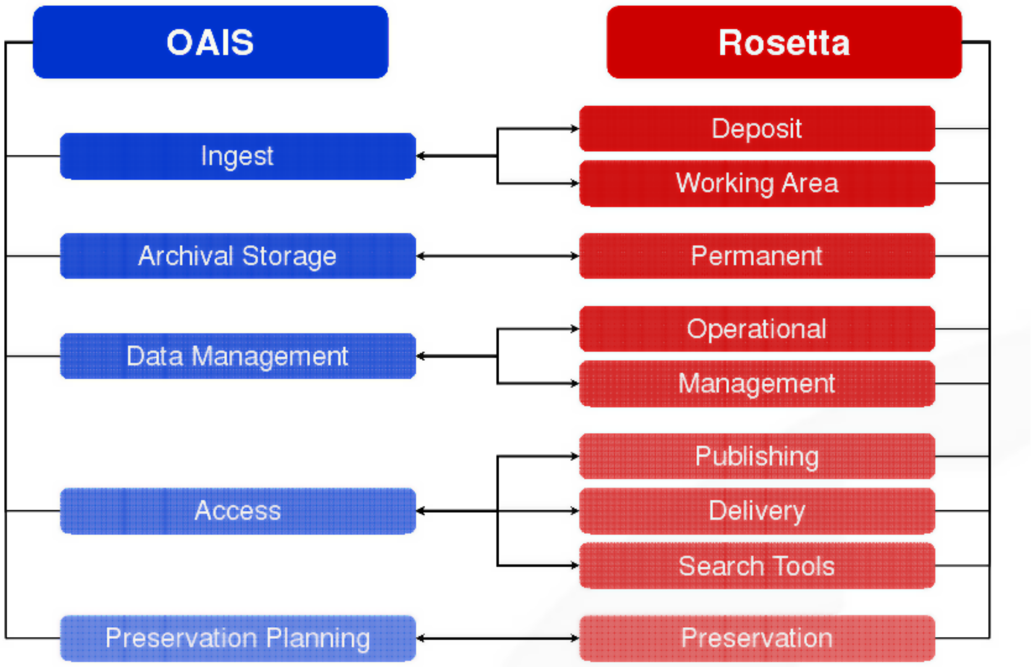

Figure 11: *Rosetta and OAIS [49]*

3 State of the Art for Submission Information Package (SIP)

3.1 Introduction to SIP

The concept of Information Package (IP) in central to the OAIS Reference Model [8]. Having a clear definition of the structure of an IP is important from the preservation point of view, because associating appropriate metadata to the content to be preserved allows its long term understanding. As shown in Figure 12, the IP is a container which links Content Information to its Preservation Description Information (PDI), i.e. metadata which enable preservation. A package is further described by Descriptive Information (also called Package Description), which enables browsing and discovery.

According to the OAIS model, the IPs are classified into:

- Submission Information Package (SIP)
  Responsible to provide the contents and the related information to the Archive during the *Ingestion* phase.
- Archival Information Package (AIP)
  Managed internally by the Archive.
- Dissemination Information Package (DIP)

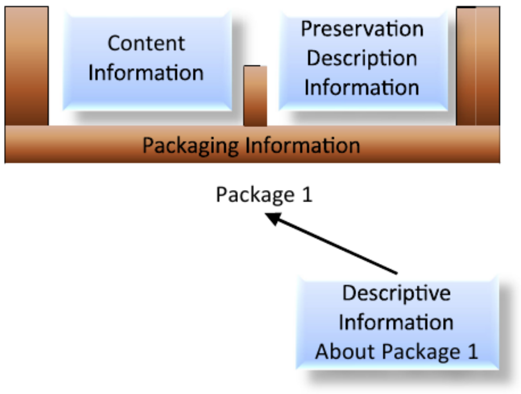
Content Information
Preservation Description Information
Packaging Information
Package 1
Descriptive Information About Package 1

Figure 12: *Information Package Concepts and Relations. From [19]*

Delivered by the Archive to the Consumer by means of the *Access* interface.

3.1.1 SIP Importance

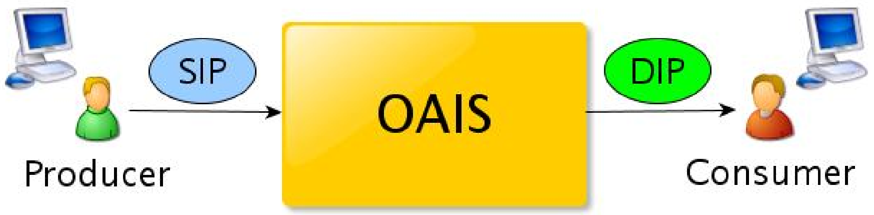

Figure 13: *SIP and DIP in relation to the OAIS Archive*

According to the OAIS Reference Model [8], SIP and DIP are the means through which an OAIS interacts with the external world, as shown in Figure 13. Therefore, SIP and DIP are fundamental for the definition of the PrestoPRIME Reference Architecture, which is OAIS compliant.

In this view, one of the objectives of this deliverable is to define the SIP structure, specifying which information is to be included, and how it is to be conveyed. OAIS Reference Model doesn't report a detailed description of what a SIP has to be, as "Its form and detailed content is typically negotiated between the Producer and the OAIS". In the model this freedom allows submitting the information needed for preservation in more than one session. According to OAIS Reference Model, a SIP may miss information without which preservation of the content is not possible. This information may be provided in a later submission session.

For PrestoPRIME SIP, some restrictions have to be placed on the definition given by OAIS

Reference Model, in order to get to a sufficiently detailed specification.

First of all, it must be clearly stated that every SIP consists of:

- *SIP wrapper*
- *metadata*
- *AV content*

Every SIP corresponds to a wrapper file, which contains references to the digital contents and their related metadata, linking the two.

A first restriction on the OAIS definition of SIP is that every SIP must correspond to one and only one Editorial Entity (EE), with the exception of SIP updates. (Note: "Editorial Entity" (EE) (PrestoPRIME) is here considered synonymous of "Intellectual Entity" (PREMIS[41]) and "Editorial Object" (PrestoSpace[51] and PrestoPRIME D5.1.1 [1]).) Another restriction is that every SIP which is not an update has to include all the metadata required for the preservation of the content. For details preservation metadata and other kinds of metadata, see PrestoPRIME Deliverable 2.2.2 [52].

3.1.2 Relationship with other Information Packages (AIP, DIP)

Defining the SIP is the first step towards the definition of the other two kinds of Information Packages. If we consider the preservation platform as a *black box*, the input documents are critical: the platform has to know in advance the structure of the objects to be handled.

Any AV content will be wrapped in a SIP at ingenstion time, in an AIP during its life inside the Archive and in a DIP when it is delivered to a Consumer. In general, for a given AV content file, SIP, AIP, and DIP will not be the same. The following three aspects have to be considered for the discussion:

1. what are the *information elements* to be included in an IP
2. what *format* is to be used to express this information
3. the specific *values* which the information elements can take

Regarding the information elements included in an IP, it can be stated that the AIP is a superset of both SIP and DIP, because it contains all the information that is included into the SIP and contains all the information necessary to construct the DIP. The AIP will contain more information than the SIP because it has to report about the preservation process, while the DIP will contain less information than the AIP because it will only contain requested parts (e.g. no reporting of preservation processes). There exists a specific use-case in which they are all the same. This is the "Take over case" in which an item is transfered (including the preservation responsibility) from one system to another. In this case, the sequence of actions is approximately the following one:

- create a DIP from AIP for export (DIP=Source:AIP)

- this DIP becomes the SIP for target system (Target:SIP=Source:DIP)
- submit the SIP

Regarding the values of the information elements, there are properties which need to be consistent with their context. For instance, the AIP will have a pointer to the data that is different from the pointer contained in the SIP. The AIP should have a link or pointer to the SIP that it was constructed from. In the same way, the DIP will point to a different location for the file (a publicly accessible one). Even in the "take over" example above many values will need to be made consistent for the subsequent recipient of the information package.

Regarding the format, SIP and DIP are defined by analogy. The definition of SIP is reported in Section 4 of this document, while for the details of the metadata format(s) see Deliverable 2.2.2 [52]. The format of an AIP can be Archive-specific. Anyway, interoperability between Archives is granted by a consistent definition of SIP and DIP.

3.1.3 SIP Creation

According to the OAIS Reference Model, the creation of the SIP is external to the Archive, under the responsability of the Producer. This creation can take place with a variable level of automation.

If, for a Producer, the frequency of submission is low, it is reasonable that SIPs will be created with manual processes, for instance through a form. The other limit case is the one in which a Producer submits many SIPs a day. In this case, for the Producer it will be worth developing a software for automatic SIP creation and submission.

The latter is a typical PrestoPRIME scenario: the Producer is the Digital Asset Management System (DAMS) running at a broadcaster's site. Probably after having broadcast a TV show, the broadcaster will make a decision to send the information to the OAIS. If the structure of information the broadcaster has in the DAMS fits the overall structure of the SIP, the DAMS will play the role of the Producer and directly submit a SIP.

3.2 SIP Examples from other Projects and Institutions

This section contains examples of SIP files provided by PrestoPRIME partners or originating from other projects or initiatives. This examples have been collected in order to know how SIPs are implemented in some OAIS inspired systems.

The first two examples are provided by ExLibris (Rosetta) and by the University of Innsbruck respectively. Both institutions make use of METS as a wrapper for the SIP. The third example is provided by the SHAMAN project, and is in the form of a RDF file. Finally, a discussion of PDPTV SIP is included.

3.2.1 Rosetta: METS and DNX

This section describes ExLibris DNX (Digital-preservation Normalized XML) format, and gives a simple example of ExLibris Rosetta System [4] SIP which makes use of DNX format. Another description, complementary to the present one, of the use METS and DNX within the Rosetta can be found in PrestoPRIME Deliverable 3.1.1 [50].

The DNX schema is a simple and unified XML schema for holding the administrative Metadata of the IE in the Permanent repository. It contains all the important data elements in a simple flat structure, divided according to the PREMIS data model and including the important technical metadata that is relevant for Preservation. The Administrative Metadata that needs to be stored arrives from various sources:

- Technical MD that is being generated by the MD extraction tools
- Access rights associated with the Material
- CMS information (system and record ID)
- Provenance information Producer, Producer Agent information, Events information.
- Miscellaneous information Links to external events, other Intellectual Entities etc.

Since all this information comes from different sources, with different standards, some of it is duplicated or organised in a way that is not useful.

Therefore the DNX profile is designed to hold all this information in a clear and organised way, with a clear mapping to the original source that allows converting it back and forth.

The DNX is written inside a METS XML file, which is created during the SIP Submission phase.

The Provenance information is written in the DNX upon moving to permanent stage since the information is still gathered along the SIP processing stage. Some of the MD is not transferred into DNX elements; it is kept as source information in the METS XML file. For example, some of the MIX information (technical MD of Image files) is not mapped into DNX records.

DNX section structure

The structure of a DNX section is as follows:

```
<section id=" Section Name ">
  <record>
    <key id="Field Name">Field Value</key>
    ...
  </record>
</section>

\end{framed}
Each record holds the fields of the section, in the form of:
\begin{framed}
\begin{Verbatim}[fontsize=\small]
<key id>Value</key>
```

As shown in the following example:

```
<dnx xmlns="http://www.exlibrisgroup.com/dps/dnx">
    <section id="generalRepCharacteristics">
       <record>
          <key id="preservationType">PRESERVATION_MASTER</key>
          <key id="usageType">VIEW</key>
       </record>
    </section>
</dnx>
```

Structure of a Repeatable Section

In case a DNX section is repeatable, there will be multiple records of the same structure as shown in the following example:

```
<section id="internalIdentifier">
  <record>
    <key id="internalIdentifierType">SIPID</key>
    <key id="internalIdentifierValue">21</key>
  </record>
  <record>
    <key id="internalIdentifierType">PID</key>
    <key id="internalIdentifierValue">REP1122</key>
  </record>
  <record>
    <key id="internalIdentifierType">DepositSetID</key>
    <key id="internalIdentifierValue">42</key>
  </record>
</section>
```

Significant Properties

In order to have a scalable structure that supports additions of technical metadata over the years, the DNX section that holds the extracted technical metadata has the following structure:

```
 <section id="significantProperties">
   <record>
     <key id="significantPropertiesType">image.maxSampleValue</key>
     <key id="significantPropertiesValue">[1]</key>
     <key id="significantPropertiesExtension"/>
   </record>
   <record>
     <key id="significantPropertiesType">image.minSampleValue</key>
     <key id="significantPropertiesValue">[0]</key>
     <key id="significantPropertiesExtension"/>
   </record>
   ...
</section>
```

This structure allows defining the technical attributes as the values of the `significantPropertiesType` field, and their value as the value of the `significantPropertiesValue` field. The `significantPropertiesType` field refers to external tables, which are part of a mechanism used in Rosetta to allow Rosetta users to configure the system and control the data.

SIP Example

The SIP described in the following is an example of a METS file in the format Rosetta is expecting while loading a SIP into the system. The specific example contains:

- IE (Intellectual Entity) information
- 'Preservation Master' Representation (File 1.gif, File 2.gif)
- 'Derivative Copy' Representation (File 1.jpg, File 2.jpg)

Streams are provided and stored outside of the METS.

In the following paragraphs, the word “Object” (capital letter) is used in the sense of the data model described in D3.1.1 [50] Section 5.1, i.e. PREMIS Object. The Data Model allows four levels of Objects, organised hierarchically, in conformance to PREMIS DD [41]:

1. Intellectual Enitity (IE)
2. Representation
3. File
4. Bistream

In the following, the words: Intellectual Enitity, Representation, File, and Bistream beginning with capital letters refer to the PREMIS entities above.

Even when the content to be preserved consists of a single file, as in the following example, in Rosetta Data Model, it is mandatory to associate to it the PREMIS-like Objects: File, Representation, Intellectual Entity.

Overview
Top-level elements of the METS file in the sample SIP are:

```
<mets:mets>
   <mets:dmdSec ID="ie-dmd">...</mets:dmdSec>
   <mets:amdSec ID="REP1122-amd">...</mets:amdSec>
   <mets:amdSec ID="FL1123-amd">...</mets:amdSec>
   <mets:amdSec ID="ie-amd">...</mets:amdSec>
   <mets:fileSec>...</mets:fileSec>
   <mets:structMap ID="REP1122-1" TYPE="PHYSICAL">...</mets:structMap>
</mets:mets>
```

File Section
The sample SIP described here comprises one TIFF file, which is associated to one File, one Representation and one Intellectual Entity Object.

The File Object is represented by a `mets:file` element with identifier `ID="FL1123"` and is associated to the administrative metadata section `ADMID="FL1123-amd"`. The Representation object associated with the File Object corresponds to the element `mets:fileGrp` with identifier `ID="REP1122"` and is associated to the administrative metadata section `ADMID="REP1122-amd"`. The IE Object is not represented in the file section. This Object is associated to the administrative metadata section with identifier `ID="ie-amd"`.

```
<mets:fileSec>
  <mets:fileGrp  USE="VIEW" ID="REP1122" ADMID="REP1122-amd">
    <mets:file ID="FL1123" MIMETYPE="image/tiff" ADMID="FL1123-amd">
      <mets:FLocat
          LOCTYPE="URL"
          xlin:href="/exlibris/.../file_1/V1-FL1123.TIF"
          xmlns:xlin="http://www.w3.org/1999/xlink"/>
    </mets:file>
  </mets:fileGrp>
</mets:fileSec>
```

Structural Map Section
The structural map section of this SIP gives no significant information.

```
<mets:structMap ID="REP1122-1" TYPE="PHYSICAL">
  <mets:div LABEL="PRESERVATION_MASTER;VIEW">
    <mets:div LABEL="Table of Contents">
      <mets:fptr FILEID="FL1123"/>
    </mets:div>
  </mets:div>
</mets:structMap>
```

Descriptive Metadata Section
There is one descriptive metadata section, which embeds Dublin Core elements. This section is conceptually related to the Intellectual Entity Object.

```
<mets:dmdSec ID="ie-dmd">
  <mets:mdWrap MDTYPE="DC">
    <mets:xmlData>
      <dc:record>
        <dc:creator>Mark Twain</dc:creator>
        <dc:title>The Adventures of Tom Sawyer</dc:title>
        <dc:type>Digitized Book</dc:type>
        ...
      </dc:record>
    </mets:xmlData>
  </mets:mdWrap>
</mets:dmdSec>
```

Administrative Metadata Section

There are three `mets:amdSec` elements, corresponding to three types of Objects.

Section `<mets:amdSec ID="FL1123-amd">` corresponds to the File Object. Section `<mets:amdSec ID="REP1122-amd">` corresponds to the Representation Object. Section `<mets:amdSec ID="IE-amd">` corresponds to the Intellectual Entity Object. Each of these sections includes DNX records appropriate to the related Object.

3.2.2 University of Innsbruck: METS

This section illustrates an example of s METS file in use at the University of Innsbruck [53] as a SIP. The example contains a scanned journal article: master files as TIFF or JPG, but also OCR files (XML) and PDFs on page level.

Overview
Top-level elements of the METS file are:

```
<METS:mets xsi:schemaLocation="..." OBJID="..." TYPE="..." LABEL="...">
   <METS:metsHdr
      CREATEDATE="2009-11-18T07:45:16"
      LASTMODDATE="2009-11-18T13:56:48"
      RECORDSTATUS="SUBMITTED">
      ...
   </METS:metsHdr>
   <METS:dmdSec ID="DCMD_ELEC">...</METS:dmdSec>
   <METS:dmdSec ID="ACV_PRINT">...</METS:dmdSec>
   <METS:amdSec ID="TECH">...</METS:amdSec>
   <METS:amdSec ID="SOURCE">...</METS:amdSec>
   <METS:amdSec ID="RIGHTS">...</METS:amdSec>
   <METS:fileSec>...</METS:fileSec>
   <METS:structMap ID="ALO_STRUCTMAP_1"
      TYPE="BOOK">...</METS:structMap>
</METS:mets>
```

METS Root element
In the METS root element the following information can be found:

1. list of referenced schemas for validation.
2. some identification metadata: `OBJID` is used as SIP-Identifier and `LABEL` with the Editorial Entity Identifier
3. attribute `TYPE` specifies the type of the content. In this case it is of type "ALO" (Austrian Literature Online)

```
<METS:mets
     xsi:schemaLocation="http://www.loc.gov/METS/
     http://www.loc.gov/standards/mets/mets.xsd
     ...
     OBJID="1016718"
     TYPE="ALO_V10"
     LABEL="AC06444458_Sozialkapital_fertig_2005">
```

Referenced schemas are: METS, Dublin Core elements, MIX, ALO, ALO_ACV.

As reported in [54], MIX (Metadata for Images in XML) is an XML schema for a set of technical data elements required to manage digital image collections. The schema provides a format for interchange and/or storage of the data specified in the Data Dictionary - Technical Metadata for Digital Still Images (ANSI/NISO Z39.87-2006). This schema is currently referred to as NISO Metadata for Images in XML (NISO MIX). [55]

ALO (Austrian Literature Online) is a digital library which comprises documents from the 11th century up to the present, whose editorial and technical responsibility is of the central computing service of the University of Innsbruck and the Department for Digitisation and Digital Preservation (DEA).[56]

See below how MIX and ALO are used in this SIP example.

Header
This elemets gives some information about SIP creation:

```
<METS:metsHdr CREATEDATE="2009-11-18T07:45:16"
    LASTMODDATE="2009-11-18T13:56:48"
    RECORDSTATUS="SUBMITTED">
    <METS:agent TYPE="ORGANIZATION" ROLE="CREATOR">
       <METS:name>Innsbruck</METS:name>
       <METS:note>This file was generated automatically</METS:note>
    </METS:agent>
</METS:metsHdr>
```

File Section

This section consists of one file group identified by `ID="MASTER"` which comprises four child file groups, identified by: `ID="JPG"`, `ID="TIF"`, `ID="XML"`, and `ID="XML"`. Each of these file groups incorporates files of the corresponding mime type. The same intellectual entity (the journal article) is stored in different formats. The article is made up of 89 image files. The order in which the pages must be read to form the whole article is specified by the attribute of the `METS:file` element `SEQ`. The first and last pages (`SEQ=1` and `SEQ=89`) are stored in JPEG format (and hence belong to the file group identified by `ID="JPG"`), while other pages are stored in TIFF format (and hence belong to the file group identified by `ID="TIF"`) (`SEQ=2` to `SEQ=89`). All the pages are stored in PDF format as well (file group identified by `ID="PDF"`).

When using ALO [57] for storing digital books (or articles), the page images are integrated into the digital repository, then from page images with a sufficient quality a raw OCR (Optical Character Recognition) text is produced. These files are those in the file group identified by `ID="XML"`).

```
<METS:fileSec>
    <METS:fileGrp ID="MASTER">
        <METS:fileGrp ID="JPG">
            <METS:file ID="JPG_00001" ADMID="ADM_JPG_00001"
            MIMETYPE="image/jpeg" SEQ="1">
                <METS:FLocat LOCTYPE="URL"
                xlink:href="http://alo-neu.uibk.ac.at/filestore/..."/>
            </METS:file>
            ....
        </METS:fileGrp>
    <METS:fileGrp ID="PDF"> ...</METS:fileGrp>
    ...
</METS:fileSec>
```

Structural Map Section

The structural map consists of one main division of `TYPE="BOOK"`.

```
<METS:div
     ID="DIV1"
     TYPE="BOOK"
     DMDID="DCMD_ELEC"
     ADMID="DCMD_ORIG DEA_RIGHTS">
   ...
```

This classifies pages (whatever the format) according to their “role” in the book. It comprises five sub-divisions:

```
<METS:div ID="DIV101" TYPE="TITLE_PAGE" ORDER="1">...</METS:div>
<METS:div ID="DIV102" TYPE="PART_GROUP" ORDER="2">...</METS:div>
<METS:div ID="DIV103"  TYPE="TOC" ORDER="3">...</METS:div>
<METS:div ID="DIV104" TTYPE="PREFACE" ORDER="5">...</METS:div>
<METS:div ID="DIV105" TYPE="PART_GROUP" ORDER="9">...</METS:div>
```

Each of these five divisions comprises further sub-divisions which associate files in different formats representing the same page, specifing the page number in the book via the attribute `ORDER`. For example, division `ID="DIV101"` is the division for the title page. It includes one (the first page only) sub-division on `TYPE="SINGLE_PAGE"` which is associated to the 3 files (different formats) which represent the first page. The attribute `ORDER="1"` establishes that, in the logical view of the structural map, this is the first page.

```
<METS:structMap ID="ALO_STRUCTMAP_1" TYPE="BOOK">
    <METS:div
    ID="DIV1"
    TYPE="BOOK"
    DMDID="DCMD_ELEC"
    ADMID="DCMD_ORIGDEA_RIGHTS">
        <METS:div ID="DIV101" TYPE="TITLE_PAGE" ORDER="1">
            <METS:div ID="DIV11" TYPE="SINGLE_PAGE"
                           ORDER="1" ORDERLABEL="">
               <METS:fptr><METS:area FILEID="JPG_00001"/></METS:fptr>
               <METS:fptr><METS:area FILEID="XML_00001"/></METS:fptr>
               <METS:fptr><METS:area FILEID="PDF_00001"/></METS:fptr>
            </METS:div>
      </METS:div>
      ...
</METS:structMap>
```

The structural map is associated to descriptive metadata section `DMDID="DCMD_ELEC"` and to administrative metadata section `ADMID="DCMD_ORIGDEA_RIGHTS"` (not present in the file).

Descriptive Metadata Section
There are two descriptive metadata sections in this METS file.

In the first one (`<METS:dmdSec ID="DCMD_ELEC">`), Dublin Core elements are embedded.

```
<METS:dmdSec ID="DCMD_ELEC">
    <METS:mdWrap MDTYPE="DC">
        <METS:xmlData>
            <dc:date>2009-11-18</dc:date>
            <dc:publisher>Innsbruck</dc:publisher>
        </METS:xmlData>
    </METS:mdWrap>
</METS:dmdSec>
```

In the descriptive metadata section `<METS:dmdSec ID="ACV_PRINT">`, ACV stands for an internal controlled vocabulary.

```
<METS:dmdSec ID="ACV_PRINT">
    <METS:mdWrap MDTYPE="OTHER" OTHERMDTYPE="ACV">
        <METS:xmlData>
            <acv:edition>Erstausgabe</acv:edition>
            <acv:fiction>Non-fiction</acv:fiction>
            <acv:dependence>Wissenschaftliche Zeitschrift
            ...
        </METS:xmlData>
    </METS:mdWrap>
</METS:dmdSec>
```

Administrative Metadata Section: Technical Metadata Sub-Section Section `<METS:amdSec ID="TECH">` comprises a technical metadata section for each file in the SIP. Technical metadata are expressed as MIX elements for images and ALO for text.

```
<METS:amdSec ID="TECH">
    <METS:techMD ID="ADM_JPG_00001">
        <METS:mdWrap MDTYPE="NISOIMG">
            <METS:xmlData>
                <mix:mix>
                    <mix:BasicImageParameters>...
                    <mix:ImageCreation>...
                </mix:mix>
            </METS:xmlData>
        </METS:mdWrap>
    </METS:techMD>
    ...
    <METS:techMD ID="ADM_XML_00001">
        <METS:mdWrap MDTYPE="OTHER" OTHERMDTYPE="ALO">
            <METS:xmlData>
                <ALO:ocr>
                    <ALO:OCRSoftware>...
                    <ALO:OCRSoftwareVersion>...
                </ALO:ocr>
            </METS:xmlData>
        </METS:mdWrap>
    </METS:techMD>
    ...
</METS:amdSec>
```

Administrative Metadata Section: Source Metadata Sub-Section Section `<METS:amdSec ID="SOURCE">` comprises one source metadata section with embedded Dublin Core elements.

```
<METS:amdSec ID="SOURCE">
    <METS:sourceMD ID="DCMD_ORIG">
        <METS:mdWrap MDTYPE="DC">
            <METS:xmlData>
                <dc:title>Das Sozialkapital....</dc:title>
                <dc:title>In:Peripherie-Zeitschrift...</dc:title>
                ...
            </METS:xmlData>
        </METS:mdWrap>
    </METS:sourceMD>
</METS:amdSec>
```

Administrative Metadata Section: Rights Metadata Sub-Section Section `<METS:amdSec ID="RIGHTS">` comprises one rights metadata section with embedded ALO elements.

```
<METS:amdSec ID="RIGHTS">
    <METS:rightsMD ID="DEA_RIGHTS">
```

```
        <METS:mdWrap MDTYPE="DC">
            <METS:xmlData>
                <ALO:allow GROUP="DEA_IP"/>
            </METS:xmlData>
        </METS:mdWrap>
    </METS:rightsMD>
</METS:amdSec>
```

3.2.3 The SHAMAN Information Package

The SHAMAN project takes the view that the OAIS Information Package needs to be extended in order to accommodate sufficient information such that hardware, software processes or terminology can be replaced without impacting the long-term use of the content. This additional information provides the context for the content by providing information on how the digital content can be preserved and understood [58]. Context information is broken-up into three components: Process Context covering the processes necessary to preserve the content as well as provenance information on how the content was produced; Domain Context covering the terminology used within the context and Enterprise Context covering the actors involved in preservation and use of the content.

Context information extends the OAIS Information package and relates to information contained in the standard OAIS packages. SHAMAN keeps the distinction between information packages used for ingestion (the SIP), archival (the AIP) and dissemination (the DIP). SHAMAN is actively working on the development of the information package and is utilising the OAI-ORE [37] as a basis for the extended information package. The first integrated prototypes that use the OAI-ORE as a basis for the SIP and AIP are currently under developement and will be published as “SHAMAN D11.2-Demonstration of Distributed Ingestion for Memory Institutions”.

The following example SIP and AIP packages do not contain context information.

The following is an example of SIP (namespace references omitted).

```
<rdf:RDF>
  <bibo:Thesis
    rdf:about="rods://shaman.cheshire3.org:1247/.../sips/1154940917475.xml">
    <dcterms:created>2010-01-07T13:17:04Z</dcterms:created>
    <ore:describes
      rdf:resource="rods://shaman.cheshire3.org:1247/.../sips/1154940917475.xml"/>
    <dcterms:creator>
      <rdf:Description rdf:about="http://foresite-toolkit.googlecode.com/#pythonAgent">
        <foaf:mbox>foresite@googlegroups.com</foaf:mbox>
        <foaf:name>Foresite Toolkit (Python)</foaf:name>
      </rdf:Description>
    </dcterms:creator>
```

```
      <rdf:type>
        <rdf:Description rdf:about="http://www.openarchives.org/ore/terms/Aggregation">
          <rdfs1:label>Aggregation</rdfs1:label>
          <rdfs1:isDefinedBy rdf:resource="http://www.openarchives.org/ore/terms/"/>
        </rdf:Description>
      </rdf:type>
      <rdf:type>
        <rdf:Description rdf:about="http://www.openarchives.org/ore/terms/ResourceMap">
          <rdfs1:label>ResourceMap</rdfs1:label>
          <rdfs1:isDefinedBy rdf:resource="http://www.openarchives.org/ore/terms/"/>
        </rdf:Description>
      </rdf:type>
      <dcterms:modified>2010-01-07T13:17:04Z</dcterms:modified>
      <dc:format>application/rdf+xml</dc:format>
      <rdfs1:seeAlso rdf:resource="urn:nbn:de:hebis:77-1082"/>
      <dc:title>960451854 Gromann, Tino Realisierung des 3_1hnHe-Kreislaufs zur ...</dc:title>
      <ore:aggregates>
        <rdf:Description
        rdf:about="rods://shaman.cheshire3.org:1247/.../files/960451854.pdf">
          <dcterms:extent>1353728</dcterms:extent>
          <pms:originalName>960451854.pdf</pms:originalName>
          <pms:hasFixity rdf:nodeID="NtvNSQwA31"/>
          <dc:format>application/pdf</dc:format>
          <xxx:hasMetadata
            rdf:resource="rods://shaman.cheshire3.org:1247/.../metadata/960451854/JHOVE.xml"/>
        </rdf:Description>
      </ore:aggregates>
    </bibo:Thesis>
    <pms:Fixity rdf:nodeID="NtvNSQwA31">
      <dcterms:created>2006-08-07T10:55:16</dcterms:created>
      <rdf:value>319d9deef8b53e375c1a76b21a48001c51fe8f71</rdf:value>
      <pms:encoding>hex</pms:encoding>
      <pms:algorithm>SHA-1</pms:algorithm>
    </pms:Fixity>
    <rdf:Description
      rdf:about="rods://shaman.cheshire3.org:1247/.../metadata/960451854/JHOVE.xml">
      <dcterms:creator rdf:resource="http://shaman.cheshire3.org/ns/demo/software/JHOVE"/>
    </rdf:Description>
  </rdf:RDF>
```

The SHAMAN SIP contains the expected metadata information such as the location of the actual content that is a candidate for archival, metadata information about the content (such as the title of the document), fixity information as well as the tools used to validate the SIP (JHOVE [59]) and extract content (the Foresite toolkit [60]).

The AIP contains a reference to the SIP that the content came from as well as fixity information and the location of the actual content in the archive. The following is an example of AIP (namespace references omitted).

```
<rdf:RDF>
  <rdf:Description
    rdf:about="rods://shaman.cheshire3.org:1247/.../metadata/960451854/JHOVE.xml">
    <dcterms:creator rdf:resource="http://shaman.cheshire3.org/ns/demo/software/JHOVE"/>
  </rdf:Description>
  <ore:ResourceMap
        rdf:about="rods://shaman.cheshire3.org:1247/.../content/aips/
        28b1974f-04b4-4637-9726-4ba7ce39e6ee.xml">
    <dcterms:modified>2010-01-08T18:09:33Z</dcterms:modified>
    <dcterms:created>2010-01-08T18:09:33Z</dcterms:created>
    <ore:similarTo
        rdf:resource="rods://shaman.cheshire3.org:1247/.../content/sips/
        28b1974f-04b4-4637-9726-4ba7ce39e6ee::1154940917475.xml"/>
    <rdf:type rdf:resource="http://purl.org/ontology/bibo/Thesis"/>
    <rdf:type>
      <rdf:Description rdf:about="http://www.openarchives.org/ore/terms/Aggregation">
        <rdfs1:label>Aggregation</rdfs1:label>
        <rdfs1:isDefinedBy rdf:resource="http://www.openarchives.org/ore/terms/"/>
      </rdf:Description>
    </rdf:type>
    <rdfs1:seeAlso rdf:resource="urn:nbn:de:hebis:77-1082"/>
    <ore:aggregates>
      <rdf:Description
        rdf:about="rods://shaman.cheshire3.org:1247/.../content/files/
        28b1974f-04b4-4637-9726-4ba7ce39e6ee/1::960451854.pdf">
        <pms:originalName>960451854.pdf</pms:originalName>
        <xxx:hasMetadata
           rdf:resource="rods://shaman.cheshire3.org:1247/.../metadata/
           28b1974f-04b4-4637-9726-4ba7ce39e6ee/1::JHOVE.xml"/>
        <xxx:hasMetadata rdf:nodeID="MQxrBAfF56"/>
        <pms:hasFixity rdf:nodeID="MQxrBAfF28"/>
        <dc:format>application/pdf</dc:format>
        <dcterms:extent>1353728</dcterms:extent>
      </rdf:Description>
    </ore:aggregates>
    <dcterms:creator>
      <rdf:Description rdf:about="http://foresite-toolkit.googlecode.com/#pythonAgent">
        <foaf:name>Foresite Toolkit (Python)</foaf:name>
        <foaf:mbox>foresite@googlegroups.com</foaf:mbox>
      </rdf:Description>
    </dcterms:creator>
    <dc:title>960451854 Gromann, Tino Realisierung des 3_1hnHe-Kreislaufs zur...</dc:title>
    <ore:describes
        rdf:resource="rods://shaman.cheshire3.org:1247/.../content/aips/
        28b1974f-04b4-4637-9726-4ba7ce39e6ee.xml"/>
    <dc:format>application/rdf+xml</dc:format>
  </ore:ResourceMap>
  <rdf:Description rdf:about="http://www.openarchives.org/ore/terms/ResourceMap">
    <rdfs1:label>ResourceMap</rdfs1:label>
    <rdfs1:isDefinedBy rdf:resource="http://www.openarchives.org/ore/terms/"/>
  </rdf:Description>
  <pms:Fixity rdf:nodeID="MQxrBAfF28">
    <rdf:value>319d9deef8b53e375c1a76b21a48001c51fe8f71</rdf:value>
    <pms:encoding>hex</pms:encoding>
    <pms:algorithm>SHA-1</pms:algorithm>
    <dcterms:created>2006-08-07T10:55:16</dcterms:created>
  </pms:Fixity>
  <rdf:Description rdf:nodeID="MQxrBAfF56">
    <dcterms:created>2010-01-07T13:17:55</dcterms:created>
    <dcterms:creator rdf:resource="http://shaman.cheshire3.org/ns/demo/software/ClamAV"/>
    <demo:virusCheckClean>True</demo:virusCheckClean>
  </rdf:Description>
</rdf:RDF>
```

The SIP and AIP examples are early prototypes and SHAMAN is developing the extended information package that includes the context information with relations to the OAIS Information Packages.

3.2.4 Preserving Digital Public Television (PDPTV): METS + PBCore + PREMIS

Preserving Digital Public Television (PDPTV) [61] is a project funded by the National Indformation and Infrasctructure Preservation Program (NDIIPP) of the Library of Congress. Projects partners are the broadcasters WNET-TV and WGBH-TV, the Public Broadcasting Service and New York University. One of the goals of PDPTV is to build a model preservation repository for digital video files. The prototype repository was built around DSpace [17], the system in use at NYU to store and manage collections of files.

SIP and AIP definition was an important stage in the developement of the prototype. The adopted metadata formats are shown in Figure 14.

- A METS file is the global wrapper for content and metadata;
- a PBCore file contains descriptive and technical metadata;
- a PREMIS file containing creating application and rendering environment information.

The resulting structure is shown in Figure 15

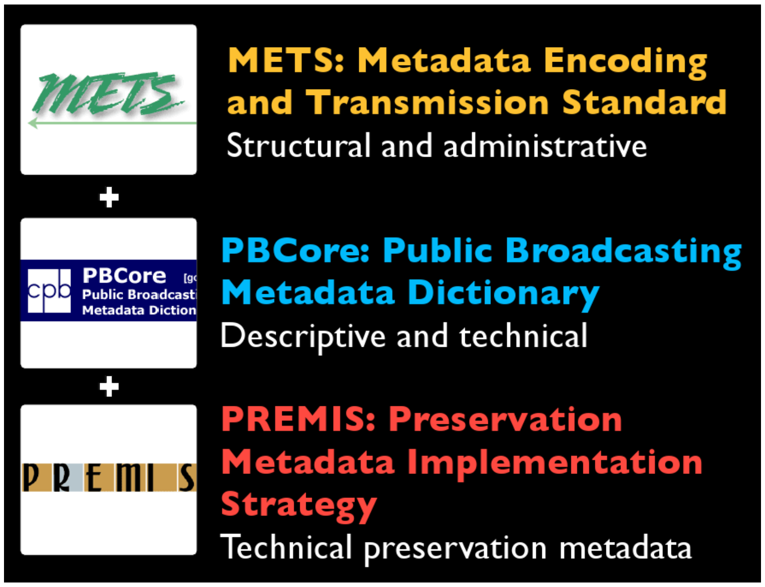
METS
METS: Metadata Encoding and Transmission Standard
Structural and administrative
+
PBCore
Public Broadcasti
Metadata Diction
PBCore: Public Broadcasting Metadata Dictionary
Descriptive and technical
+
PREMIS
PREMIS: Preservation Metadata Implementation Strategy
Technical preservation metadata

Figure 14: *PDPTV Metadata Schema(from [62])*

Further details on PDPTV history, repository design and SIP/AIP definition can be found in [63] and [64].

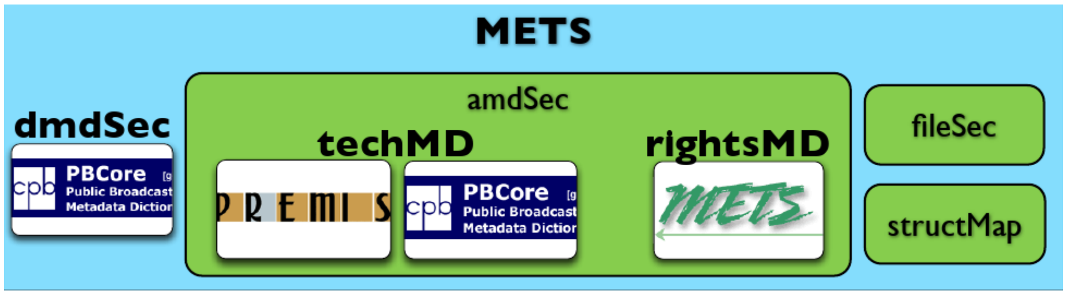

Figure 15: *Structure of PDPTV SIP (from [62])*

4 Submission Information Package (SIP): Wrapper Specification

In this Section, the following points will be specified for the PrestoPRIME Preservation Platform SIP:

1. the list of supported AV formats;
2. the list of allowed structural relationships between AV content files within one SIP;
3. the list of metadata information to be included (mandatory, recommended or optional). No restrictions are placed on the format;
4. the SIP wrapper format;
5. how information elements are to be inserted in the wrapper.

It is important to highlight some basic assumptions valid for every SIP, with the exception of SIP updates:

1. each SIP must be preservable, i.e. contain all the information needed for preservation;
2. each SIP corresponds to one Editorial Entity.

4.1 Audiovisual Content

In the professional and semi-professional AV-domain the wrapper formats which are currently in use for the master and higher quality levels are:

- Generic Wrapper (can accomodate various types of codecs)
  - MXF [65]
  - AVI [66]

  - MOV/Quicktime [67]
- Non generic wrappers (strictly related to one codec or a family of codecs)
  - RAW DV
  - MPEG multiplex (subtypes are TS, PS, MP4)
  - other

MXF, specified by SMPTE [68], was specifically intented for overcoming the dualism between Quicktime and AVI and the resulting interoperability problems.

This objective is currently only partially achieved because:

- MXF is defined by a wide set of specifications, with many resulting variants for which various interoperability issues were raised. In particular ten Operational Patterns are defined in addition to the specifications defining the Mapping of each content type with the generic essence container specification.
- The other formats, in particular QuickTime, are still popular because they provide some benefits in specific contexts.

Besides there is the need to consider that small archives may have material in formats which would be considered of lower quality level by the larger organisations and that the MXF format is still almost unknown out of its reference community. (Elements useful for deepen this discussion can be found in PrestoPRIME Deliverable 5.1.1 [1] from which input was also taken).

The reference architecture specification requires that only the formats registered and described in the “registry of supported AV formats” will be considered valid. A specific implementation may provide support to any format, provided the availability of the necessary tools.

Formats mandatory or recommended for PrestoPRIME compliant implementation are listed in Tables 1 and 2. The first column is an abbreviation of the format which will be used as an identifier. The second column specifies the Operational Pattern under consideration (if applicable). The third column specifies the video essence container, and the fourth column the audio essence container. The fifth column specifies the support level for the format in the open reference implementation. The last column reports the output of `mxfdump` for the Essence Container. `mxfdump` is a command which is part of MXFLIB package [69]. It is a possible means to identify the format of an MXF file.

Table 1 lists the supported formats when they come in the form of an MXF Operational Pattern 1a. In general, MXF OP1a files contain both audio and video, but it’s possible to have only video or only audio, even though this is a special case. The audio and video tracks can be included in the same essence container (as in the case of D10), or in separate essence containers (this is the case of multiple mappings).

In the PrestoPRIME, framework the reference test case content wrapping format is MXF-OP1A, although further formats are supported by the “reference architecture specification” and by the “open reference implementation”.

Table 2 lists the supported formats when they come in the form of a set of MXF Atom files. An Atom Set is made of various files each of which is either only one video or only one audio channel. These files have to be played at the same time.

Note: in the following paragraphs the term “Atoms” is used to refer to any single component file, and must not be confused with the particular Operational Pattern of MXF *OPAtoms*. To avoid confusion:

- to refer to any single-component file, the term “Atom” will be used;
- to refer MXF Atoms, the expression “OPAtoms” will be used.

The “OPAtoms” are thus a special case of “Atoms”.

The operational patterns reported in the tables are only OPAtoms or OP1a. More complicated Operational Patterns will not be considered in this specification. It was not possible to complete the last column for the mxf_vc3 and mxf_vc3_atoms because of the lack of a sample file on which to apply `mxfdump`.

Table 1: *Supported formats: OP1A*

Format Abbreviation	**OP**	**Video**	**Audio**	**Mandatory or Recommended**	**Essence Container(s) (output on `mxfdump`)**
mxf_d10	OP1a	D10	AES	MANDATORY	`MXF-GC SMPTE D-10 Mappings`

mxf_mpeg	OP1a	MPEG	AES	MANDATORY	Multiple mappings:
					`MXF-GC MPEG Elementary Streams`
					`MXF-GC AES-BWF Audio`
					mxf_xdcam is a subset of this

mxf_aes	OP1a	—	AES	MANDATORY	`MXF-GC AES-BWF Audio`

mxf_j2k	OP1a	JPEG2000	AES	MANDATORY	Multiple mappings:
					`MXF-GC JPEG-2000 Picture Mappings`
					`MXF-GC AES-BWF Audio`

mxf_unc	OP1a	Uncompressed Pictures	AES	MANDATORY	Multiple mappings:
					`MXF-GC Uncompressed Pictures`
					`MXF-GC AES-BWF Audio`

mxf_dv	OP1a	DV-DIF	AES	RECOMMENDED	Multiple mappings:
					`MXF-GC DV-DIF Mappings`
					`MXF-GC AES-BWF Audio`

mxf_vc3	OP1a	VC3	AES	RECOMMENDED	Multiple mappings:

Table 2: *Supported formats: MXF Atoms*

Format Abbreviation	**OP**	**Video**	**Audio**	**Mandatory or Recommended**	**Essence Container(s) (one for each file)**
mxf_d10_atoms	Atoms Set	D10	AES	OPTIONAL	`MXF-GC SMPTE D-10 Mappings`
mxf_mpeg_atoms	Atoms Set	MPEG	AES	OPTIONAL	`MXF-GC MPEG Elementary Streams`
mxf_aes_atoms	Atoms Set	—	AES	OPTIONAL	`MXF-GC AES-BWF Audio`
mxf_j2k_atoms	Atoms Set	JPEG2000	AES	OPTIONAL	`MXF-GC JPEG-2000 Picture Mappings`
mxf_unc_atoms	Atoms Set	Uncompressed Pictures	AES	OPTIONAL	`MXF-GC Uncompressed Pictures`
mxf_dv_atoms	Atoms Set	DV-DIF	AES	OPTIONAL	`MXF-GC DV-DIF Mappings`
mxf_vc3_atoms	Atoms Set	VC3	AES	OPTIONAL	

Possible relationships betweeen the AV-media files submitted in a SIP are listed in Tables 3 and 4. It is required for a SIP to be compliant to one of the defined structures.

In each of the cases reported in the tables, there is one SIP, corresponding to one Editorial Entity (EE). This EE can come in one or more copies, each one in the form of one or more AV files.

The simplest relationship is *One-to-one*: a single AV file for the realisation of the EE. This is the default case for submission.

A more complex situation is that in which more than one AV file are required for the realisation of the EE. One possiblity is the realisation of the EE through a group of single component files (audio or video) which have to be played at the same time to render the EE. This is the *One-Atoms* case. This means that the complete material is recorded within multiple files, which are deemed to have the same duration. This approach is valid for non-MXF files as well as MXF files.

Another possibility for the structure of AV files in one SIP is that of files which have to be played in sequence. This is the *One-Sequence* case. Other cases are combinations of the simpler cases above.

The structures listed in the tables are illustrated in Figure 16 and Figure 17.

Table 3: *Possible structures of relationships betweeen SIP and the AV-media files to be submitted*

Structure name	**Allows more than one copy (different quality levels)**	**Temporal sequence of files**	**Single component files (Atoms)**	**Support for PPRIME compliant implementation**	**Constraints and notes**
One-to-One	NO	NO	NO	MANDATORY	
One-Atoms	NO	NO	YES	RECOMMENDED	All component files MUST have the same play duration. All the files of the same component type MUST be of the same format.
One-Sequence	NO	YES	NO	RECOMMENDED	It must be specified if overlaps are present or not. In the latter case the files may be played in sequence. All the files MUST be of the same format.
One-Atoms-Sequence	NO	YES	YES	OPTIONAL	This is a sequence of Atoms. For each Atom all component files MUST have the same play duration. It must be specified if overlaps are present or not. In the latter case the file groups may be played in sequence. The number of component files MUST be the same for all the Atoms in sequence.

Table 4: *Possible structures of relationships betweeen SIP and the AV-media files to be submitted*

Structure name	**Allows more than one copy (different quality levels)**	**Temporal sequence of files**	**Single component files (Atoms)**	**Support for PPRIME compliant implementation**	**Constraints and notes**
Multi-one	YES	NO	NO	OPTIONAL	Each copy is a One-to-One
Multi-Atoms	YES	NO	YES	OPTIONAL	Each copy is either a One-to-One or a One-Atoms. At least one copy is One-Atoms
Multi-Sequence	YES	YES	NO	OPTIONAL	Each copy is either a One-to-One or a One-Sequence. At least one copy is One-Sequence
Multi-Any	YES	YES	YES	OPTIONAL	Each copy can be any of single-copy structures.

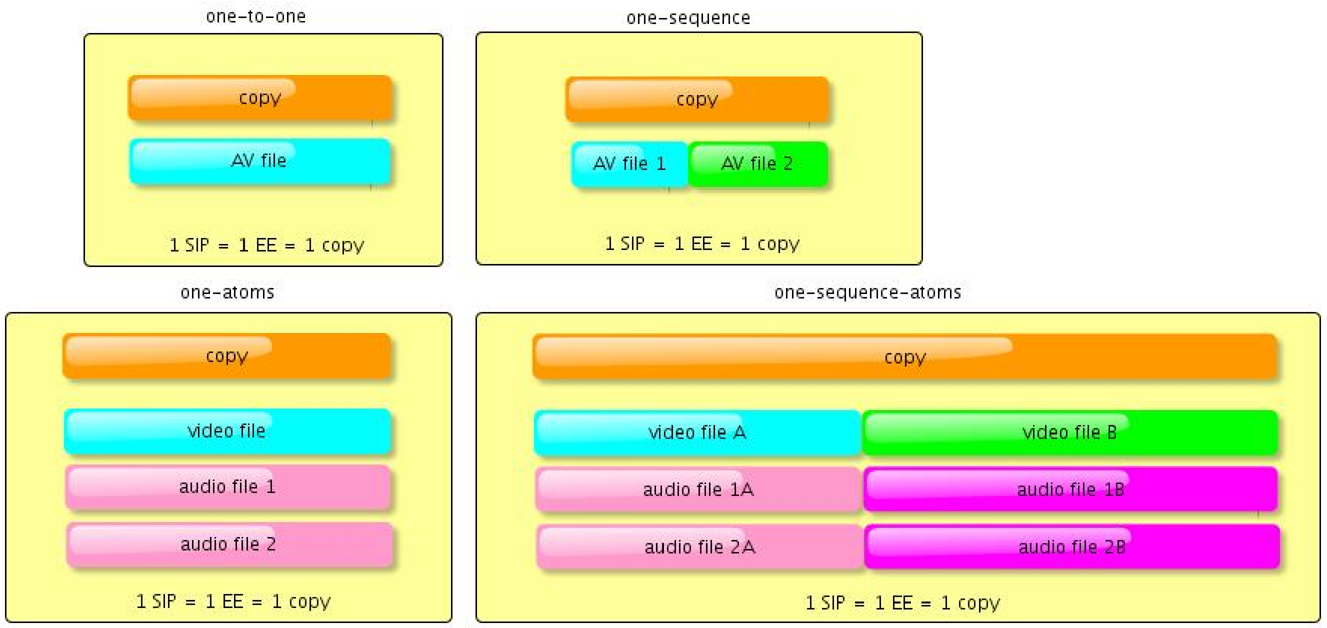

Figure 16: *Structures that allow one copy.*

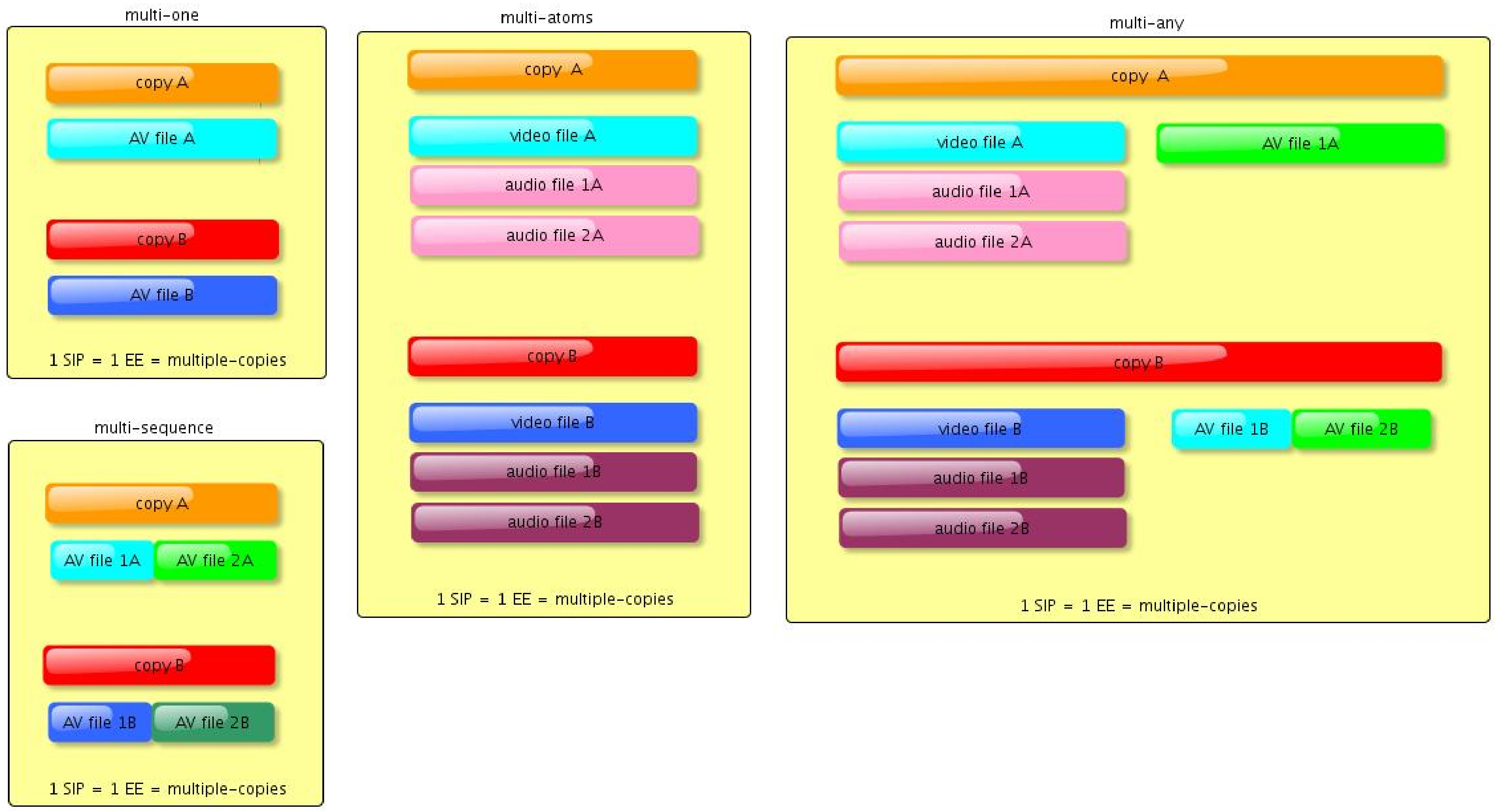

Figure 17: *Structures that allow multiple copies.*

4.2 Metadata

A classification of metadata included in the SIP wrapper is reported in Tables 5 and 6. Each of the listed groups of metadata is either mandatory or recommended in a SIP. Deliverable 2.2.2 [52] provides an overview and analysis of the different candidate formats for each group of metadata properties. Deliverable 2.1.3 [70] will document the decisions for the metadata formats supported in the PrestoPRIME data model.

Each metadata type has a requirement level, which is one of:

- *mandatory* - this information is required for SIP ingestion to be successful, otherwise the SIP must be rejected (as it would cause errors to occur in the system)
- *recommended* - these are *optional* elements, the lack of which does not cause rejection (no errors caused), but the coverage of scenarios would be poor
- *optional* - elements not essential in the coverage of the scenarios (but might be quite important for specific context)

Section 4.3 specifies in which point of the SIP wrapper each kind of metadata has to be placed.

Table 5: *Metadata included in PrestoPRIME SIP, requirement level, update option*

Information	**Requirement Level**	**Updatable**	**Comments and Examples**
Identification metadata	mandatory	yes/no	Information for indentifying the Editorial Entity updatable: title, credits, etc. non-updatable: actual Editorial Entity identifiers
Descriptive metadata	optional	yes	
Technical metadata	recommended		If present implies verification on ingestion, which would permit the detection of discrepancies with the provided information. If not present the technical metadata will be extracted by direct inspection of AV material and thus no discrepancy check is possible. This information is required to interpret the content. It can be duplicated in the content wrapper.
Rights metadata	recommended	yes	The digital rights metadata is provided by one owl file. The definition of the rights metadata is under the responsibility of WP4T4.

Table 6: *Metadata included in PrestoPRIME SIP, requirement level, update option*

Information	**Requirement Level**	**Updatable**	**Comments and Examples**
Ingest and Preservation Options and Service Terms	mandatory	yes	E.g. • permission/request to create a browsing quality copy • permission/request to create access copy • permission to change the file format • reference to a specific defined and agreed policy • preservation and delivery SLA
Provenance	mandatory, but history previous to ingestion: optional	yes	Some initial provenance information must be supplied by the Producer, such as: • which Producer and which ingest flow the content comes from and when it was ingested (mandatory) • previous history (optional) The Archive will update the history section when a relevant event happens to the file within the Archive itself.
Update and access permissions	mandatory	yes	• group of users with update permission (content/rights/metadata/policy) • group of users with permission to search & browse (content/rights/metadata/policy) • group of users with access to delivery (quality level) User profiles need to be created.

4.3 Wrapper: METS

METS has been chosen as the overall container for the ingestion phase because of the offered flexibility and ease of implementation, stated by the experience of several partners.

METS (Metadata Encoding and Transmission Standard) [71] is a standard information container developed by the Library of Congress. METS files are designed to comprise complex content, so there is no need to have a one to one correspondence between METS files and content files. Instead, an intellectual unit (e.g. TV show) can be a METS unit and contain all metadata and all content files (different ones, like pages of a book, or volumes of a journal).

An analysis of METS structure version 1.9 (specified in [72]). Figure 18 illustrates METS sections.

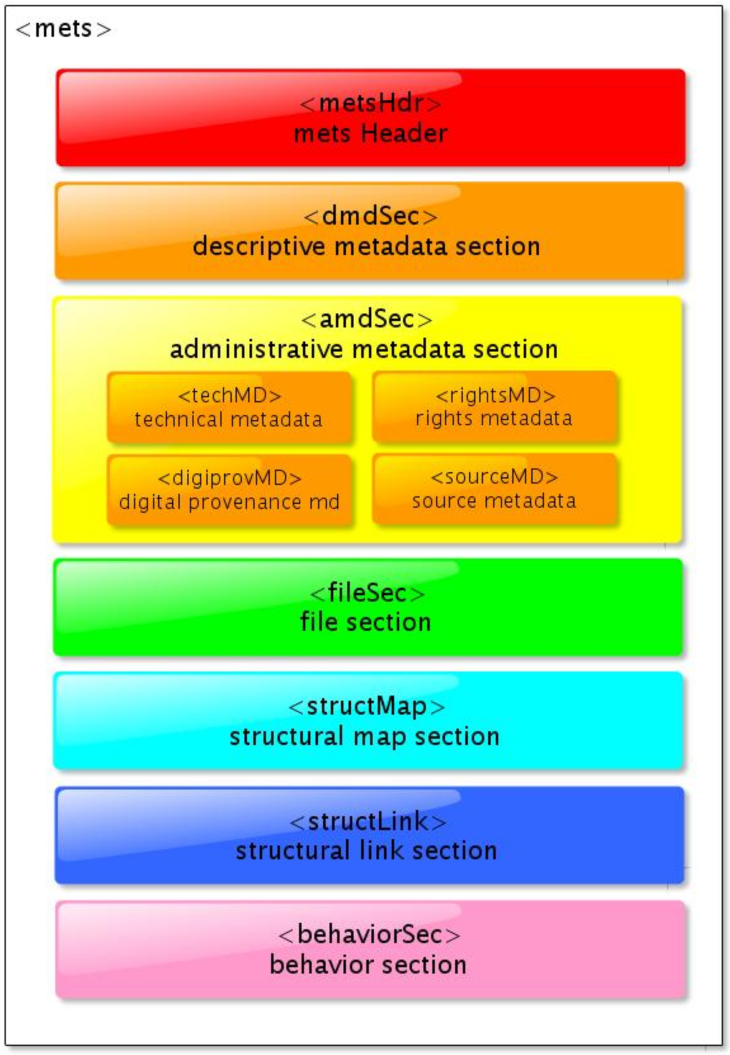

Figure 18: *METS sections*

A description of how requirements of Tables 5 and 6 fit in this structure can be found in the following paragraphs.

Each of the elements of the list of requirements outlined in Tables 5 and 6 is associated to the METS element which seems the most appropriate to express it. In the following, *emphasised* words are used to refer to the list of requirements of Tables 5 and 6.

For every SIP there is one corresponding Editorial Entity (EE). The default case for submission is one SIP to one EE to one AV-media file (master level) (One-to-one case).

General mechanisms

METS offers two mechaninsms for conveying information:

1. Embedding information
   It is possible to embed non-METS-XML code, as well as non-XML text, or even binary. XML should be recommended as the System will understand only information provided according to the agreed format. Here's an example of embedded Dublin Core elements:

```
<mets:mdWrap MDTYPE="DC">
  <mets:xmlData>
   <dc:record>
      <dc:title>My favourite Tv show</dc:title>
   </dc:record>
  </mets:xmlData>
</mets:mdWrap>
```

2. Referencing information
   In a preservation context, it is assumed that such information is loaded and becomes part of the preserved set. Example:

```
<mets:mdRef
   LOCTYPE="URL"
   MIMETYPE="application/xml"
   MDTYPE="OTHER"
   OTHERMDTYPE="ANOTHERTYPE"
   LABEL="description"
   SIZE="123"
   CHECKSUM="0123456789abcdef"
   CHECKSUMTYPE="MD5"
   CREATED="2010-02-17"
   xlink:href="//ourplace.org/myinfo001.xml"/>
```

Either of these mechanisms can be used in the following METS sections:

- `dmdSec`
- `rightsMD`
- `sourceMD`
- `digiprovMD`
- `techMD`

Another mechanism of which METS makes extensive use is assigning internal identifiers to each section. It is possible (and often required) to make reference from one section to another one by that ID.

METS Root Element

The root element of a METS file is `<mets:mets>`. It is intended for:

1. Specification of reference schemas for validation
2. State if the SIP is an update or not

The root element attribute `LABEL` should be used to identify the SIP as a:

- PrestoPRIME SIP (non-update): `LABEL="pPRIME SIP"`;
- update SIP: `LABEL="pPRIME SIP-UPDATE"`

The attribute `OBJID` is is the SIP identifier. This attribute is mandatory. It is allowed to take a value only ONCE.

Example:

```
<mets:mets
    xmlns:mets="http://www.loc.gov/METS/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.loc.gov/METS/ mets.xsd"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    LABEL="pPRIME SIP" [or "pPRIME SIP-UPDATE"]
    OBJID="SIP001"  >
</mets:mets>
```

Header

This element is optional. It provides information about the SIP creation (when/who). Information reported is identification metadata for the METS document itself, not for the EE. Example:

```
<mets:metsHdr
   CREATEDATE="2010-02-16T15:00:00"
   RECORDSTATUS="Complete" >
      <mets:agent ROLE="CREATOR" TYPE="ORGANIZATION">
          <mets:name>PrestoPRIME</mets:name>
      </mets:agent>
      <mets:agent ROLE="OTHER" TYPE="ORGANIZATION">
          <mets:name>PrestoPRIME</mets:name>
      </mets:agent>
</mets:metsHdr>
```

Descriptive Metadata Section

This section includes the Editorial Entity *identification metadata*. The use of an embedded `dc:identifier` element is recommended. Multiple identifiers can appear.

The following snippet is an example of use of embedded Dublin Core metadata. Besides the recommended `dc:identifier` element, other Dublin Core elements might be useful for identification of the Editorial Entity (Title, Identifier, Creator, Contributor, Publisher).

```
<mets:dmdSec ID="dmd002">
  <mets:mdWrap MDTYPE="DC">
    <mets:xmlData>
      <dc:record xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema- instance">
         <dc:identifier>urn:pprime:rai#014</dc:identifier>
         <dc:title>My favourite Tv show</dc:title>
      </dc:record>
    </mets:xmlData>
  </mets:mdWrap>
</mets:dmdSec>
```

Alternatively, an external resource can be loaded using the `mdRef` mechanism.

All the *descriptive metadata* have to be placed in this section. This information in either embedded or referenced as usual. An example is the “Description” and the “Subject” elements in a Dublin Core record.

The embedding mechanism shall allow to submit metadata according to formats according to the producers policies. The format may or may not be supported by the preservation system, depending on agreements.

Formats considered for inclusion are listed in D4.0.1 Part A, Section 2.2, considering as candidate formats: MPEG-7 DAVP, MPEG-7 ECM SCAIE, P_Meta, PrestoSpace EDOB, DC/DCterms, EBUcore, PB.

Administrative Metadata Section

This section allows four subsections:

1. `sourceMD` - to record information about the source media/format - In the case of AV it can be about the media before digitisation
2. `digiprovMD` - to record any preservation-related actions
3. `rightsMD` - to record *rights metadata*
4. `techMD` - to record *technical metadata*

Technical Metadata Section This section is used to provide *technical metadata*. As usual, two mechanisms are possible:

1. XML data within the METS document

```
<mets:amdSec ID="amd001">
   <mets:techMD ID="amd001tech001">
       <mets:mdWrap MDTYPE="OTHER" OTHERMDTYPE="anothertype">
           <mets:xmlData><anothertypemetadata/></mets:xmlData>
       </mets:mdWrap>
   </mets:techMD>
   ...
</mets:amdSec>
```

2. Reference to external file to be loaded.

```
<mets:amdSec ID="amd001">
   <mets:techMD ID="amd001tech001">
       <mets:mdRef
         ...
         xlink:href="//mystuff.org/mytech001.xml"/>
   </mets:techMD>
</mets:amdSec>
```

It is recommended to have one section for each AV-media file. The information contained here must be coherent with the information obtainable by direct inspection of the AV-media file.

The *technical metadata* is to be included here. The list of information elements (i.e. the vocabulary) may be taken from EBU Tech 3295 TechnicalDetails, or SMPTE dictionary (RP-210), or a mixture of both. Both mentioned vocabularies have XML representations, but it has to be defined which one to use for potentially needing to express properties from both dictionaries. Neither METS nor PREMIS define a generic way of including metadata properties except for including XML elements in any schema. To make use of elements from both P_Meta and SMPTE RP201, P_Meta XML and SMPTE 434M formats have to be mixed.

The `techMD` is the section where the *Ingest and Preservation Options and Service Terms* are to be included. This section will probably make use of PREMIS (or DNX as a subset of PREMIS), maybe with some extensions for AV specific options.

Update and access permissions also find their place in this section. At present time the format for these metadata is still to be discussed.

Rights Metadata Section This METS section obviously corresponds to the *rights metadata* mentioned in the requirements Tables 5 and 6.

Digital rights information is provided by reference to an external OWL file (for further details see outputs of WP4T4) to be loaded (file attributes are omitted in the examples below).

The handling of that information within the preservation system will be run by ad-hoc services. Thats why the supported mechanism for submission is that of loading external resources.

```
<mets:rightsMD ID="amd001rights-owl" GROUPID="rights001">
   <mets:mdRef
   LOCTYPE="URL"
   MIMETYPE="application/rdf+xml"
   MDTYPE="OTHER"
   LABEL="our rights on content"
   OTHERMDTYPE="pprime-rights"
   xlink:href="https://ourstuff.org/myrights001.owl"/>
</mets:rightsMD>
```

Digital Provenance Section METS specification describes this section as the section in which any preservation related event is to be recorded. Hence, for a SIP about to be ingested it will only contain information about the Producer of the content and about digital preservation actions previous to the submission to the Archive. This is the section which incluces *Provenance* information. At present time the format for these metadata is still to be discussed. For further details, see the Conclusions section of Deliverable 2.2.2 [52].

Note: the term “Provenance” can be interpreted as meaning metadata as the information explaining how an AV-entity has been created by the reuse of pre-existing AV material. In this case, it may be appropriate to place this information in the `mets:dmdSec`. It might be useful for the Archive to know that the Editorial Entity A is made of parts coming from a list of other Editorial Entities. In such a case, the material instances realising the Editorial Entities have to be identified and the information related to durations and timepoints on the material timeline has to be included.

File Section

This section contains the list of links to the AV-media files to be submitted. This list is unstructured as relations between possible multiple files are expressed in the Structural Map section.

```


<mets:fileSec>
   <mets:fileGrp >
      <mets:file ID="f001" ADMID="t001">
         <mets:FLocat LOCTYPE="URL"
             xlink:href=
             "https://fileportal.rai.it/RAI_014.d10.mxf"/>
         </mets:file>
         <mets:file ID="f002" ADMID="t002">
             <mets:FLocat LOCTYPE="URL"
             xlink:href=
             "https://fileportal.rai.it/RAI_014.ogv"/>
```

```
          </mets:file>
    </mets:fileGrp>
</mets:fileSec>
```

Structural Map Section

For each SIP, there MUST be one and only one `structMap` element.

The Structural Map element must be used according to PrestoPRIME AV structures specified in Section 4.1.

The number of `structMap` elements related to AV content is limited to one element, which MUST have the attribute `LABEL=``PPRIME-AV''`.

The attribute `TYPE` is used for indicating the structure of the submitted AV-media files. The values are those given in the column “Structure name” of Table 7.

The number of levels of the hierarchy of `div` elements is at most four, depending on the type of structure.

For the default case (One-to-one), this section has a simple form:

```
<mets:structMap LABEL="PPRIME-AV" TYPE="One-to-One">
       <mets:div TYPE="AudioVisual">
           <mets:fptr FILEID="f001"/>
       </mets:div>
</mets:structMap>
```

The attribute `TYPE` of `div` elements MUST be present, with a value according to Table 8.

When there is a `div` of `TYPE="Sequence"`, its `div` sub-elements MUST have the attribute `ORDER`. The value of the attribute `ORDER` is the position of the `div` sub-element in the sequence. The `div` sub-elements can be leaves or not. Example:

```
<div TYPE="Sequence">
  <div TYPE="AudioVisual" ORDER="1"><fptr FILEID="f001"/></div>
  <div TYPE="AudioVisual" ORDER="2"><fptr FILEID="f002"/></div>
</div>
```

Table 7: *Number of levels of the hierarchy of* `div` *elements depending on the type of structure.*

Structure Name	**Number of `div` levels**
One-to-One	1
One-Atoms	2
One-Sequence	2
One-Atoms-Sequence	3
Multi-One	2
Multi-Atoms	3
Multi-Sequence	3
Multi-Any	4

Table 8: *Values of the attribute `TYPE` of `div` elements*

Value of `TYPE` for `div` element	**Meaning**	**Constraints**
AudioVisual	Related material is a multiplex, possibly including audio, video, and data (e.g. subtitles) components	MUST be leaf `div` element Cannot be child of `div` with `TYPE=''Atoms''`
Video	Related material is video only	MUST be leaf `div` element
Audio	Related material is audio only	MUST be leaf `div` element
Data	Related material is data only	MUST be leaf `div` element
Atoms	There is a set of materials, single component, with the same duration.	Cannot be leaf Child `div` elements MUST be of `TYPE` Video, Audio, or Data
Sequence	There is an ordered set of materials. Overlaps are permitted	Cannot be leaf Child `div` can be either of `TYPE` Atoms or a leaf Child `div` MUST use the attribute `ORDER`
Sequence without Overlaps	There is an ordered set of materials, without overlap	Cannot be leaf Child `div` can be either of `TYPE` Atoms or a leaf Child `div` MUST use the attribute `ORDER`
MultiCopy	There is a set of copies of the same content.	Cannot be leaf MUST be top level (child of `structMap`)

When there is a div of TYPE="Atoms", its div sub-elements MUST have the attribute LABEL. The value of the attribute LABEL has the purpose of distinguishing between files of the same format. The utility of this mechanism becomes evident when one wants to represent the structure One-Atoms-Sequence (or any more complicated structure which has this as a sub-structure). For instance, in the following example:

```
<structMap  LABEL="PPRIME-AV" TYPE="One-Atoms-Sequence">
   <div TYPE="Sequence">
      <div TYPE="Atoms" ORDER="1">
         <div TYPE="Video" LABEL="video"><fptr FILEID="f001"/></div>
         <div TYPE="Audio" LABEL="left"><fptr FILEID="f002"/></div>
         <div TYPE="Audio" LABEL="right"><fptr FILEID="f003"/></div>
      </div>
      <div TYPE="Atoms" ORDER="2">
         <div TYPE="Video" LABEL="video"><fptr FILEID="f004"/></div>
         <div TYPE="Audio" LABEL="left"><fptr FILEID="f005"/></div>
         <div TYPE="Audio" LABEL="right"><fptr FILEID="f006"/></div>
      </div>
  </div>
</structMap>
```

the attribute LABEL lets one know that file f005 is the continuation of file f002 and file f006 is the continuation of file f003.

When there is a div of TYPE="MultiCopy", ONE of its div sub-elements MUST have the attribute LABEL="Master". This labels one of the copies as the master copy. Example:

```
<structMap  LABEL="PPRIME-AV" TYPE="Multi">
   <div TYPE="MultiCopy">
      <div TYPE="AudioVisual" LABEL="Master">
          <fptr FILEID="f001"/>
      </div>
      <div TYPE="AudioVisual">
          <fptr FILEID="f002"/>
      </div>
   </div>
</structMap>
```

Structural Link Section

Not in use.

Behavior Section

Not in use.

4.4 Specification Summary and Match of the Requirements to METS elements

Every PrestoPRIME SIP consists of: AV content, metadata, and SIP wrapper.

The SIP wrapper is METS [71].

Supported content file formats are listed in Tables 1 and 2. The reference test case for the submitted AV content is one MXF OP1a D10 file. This is deemed to be a “typical case” as identified within the Scenarios (D5.1.1).

Required metadata scoring against the METS structure is given in Table 9 and illustrated in Figure 19.

The section `fileSec` consists of a listing of the AV files to be ingested. If there’s more than one file, the files must combine in some way to render the Enditorial Entity associated with the SIP (remind: one SIP corresponds to one EE). The way files have to be played to render the EE is expressed by the Structural Map, as specified in Section 4.3.

4.5 Simple Examples of Use of METS

In this Section, some stub examples of SIP are reported. Namespace references are omitted.

1. One-to-one - the first example is the default case: one SIP to one Editorial Entity to one AV-media file (master level).

   The exceptions to the default case which will be considered, from highest to lowest priority, are:

2. Update SIP - only for rights in the example, editorial identification and descriptive information update - NO AV-media file

3. One-Sequence - one SIP to one Editorial Entity to an ordered sequence of AV-media files (master level)

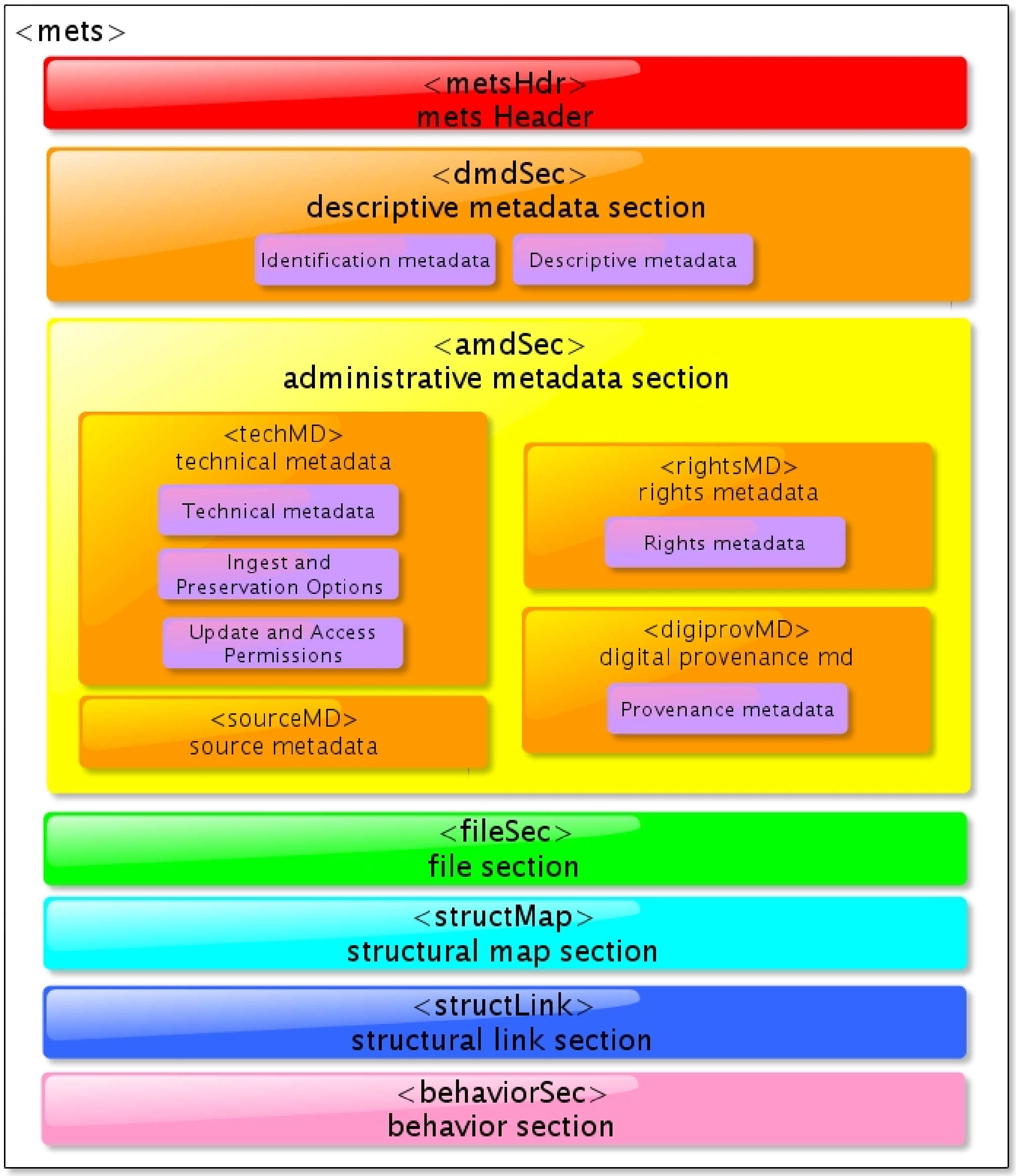

Figure 19: *Mapping of rows of Tables 5 and 6 to METS elements*

Table 9: *Mapping of rows of Tables 5 and 6 to METS elements*

Information	**Corresponding METS element**
Identification metadata	The identification metadata for the Editorial Entity is to be placed in the Desriptive Metadata Section: identification information can be conveyed in any metadata format, either embedded in or referenced from `dmdSec`. The use of `dc:identifier` is recommended.
Descriptive metadata	Descriptive Metadata Section
Technical metadata	Administrative Metadata Section - Technical Metadata subsection
Rights metadata	Administrative Metadata Section - Rights Metadata subsection
Ingest and Preservation Options and Service Terms	Administrative Metadata Section - Technical Metadata subsection
Provenance	Administrative Metadata Section - Digital Provenance subsection
Update and access permissions	Administrative Metadata Section - Technical Metadata subsection

4. One-Atoms - one SIP to one Editorial Entity to a compound of single component AV-media files (master level)
5. One-Atoms-Sequence - one SIP to one Editorial Entity to a sequence of groups of Atoms
6. Multi-One - one SIP to one Editorial Entity to multi format AV-media files (one master and various other quality levels)

4.5.1 Default case: One-to-one

```
<mets>
    <dmdSec ID="d001">
      <mdWrap MDTYPE="DC">
         <xmlData>
            <dc:identifier>urn:pprime:rai#014</dc:identifier>
         </xmlData>
      </mdWrap>
   </dmdSec>
   <amdSec ID="a001">
      <techMD ID="t001">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <rightsMD ID="r001">
         <mdRef LOCTYPE="URL" MDTYPE="OTHER"
             xlink:href="pprai014.owl"/>
      </rightsMD>
   </amdSec>
   <fileSec>
      <fileGrp>
         <file ID="f001" ADMID="t001">
            <FLocat LOCTYPE="URL"
            xlink:href="https://fileportal.rai.it/RAI_014.d10.mxf"/>
         </file>
      </fileGrp>
   </fileSec>
   <structMap  LABEL="PPRIME-AV" TYPE="One-to-One">
      <div TYPE="AudioVisual">
         <fptr FILEID="f001"/>
      </div>
   </structMap>
</mets>
```

4.5.2 Case 2: Update

This case has the following characteristics:

1. The value of `LABEL` identifies the Editorial Entity to update
2. The value of `PROFILE` specifies it's an update
3. File Section is missing
4. Administrative, Technical, Source, and DigiProv sections are missing
5. Structural Map is empty but present because mandatory in METS

```
<mets:mets
  OBJID="002"
  LABEL="Sample 001"
  PROFILE="pPRIME SIP-UPDATE"
  TYPE="AudioVisual">
    <mets:metsHdr
      CREATEDATE="2010-02-17T15:00:00"
      RECORDSTATUS="Complete" >
        <mets:agent
           ROLE="CREATOR"
           TYPE="ORGANIZATION">
              <mets:name>PrestoPRIME</mets:name>
        </mets:agent>
    </mets:metsHdr>
    <mets:dmdSec ID="dmd001">
       <mets:mdRef
          LOCTYPE="URL"
          ...
          xlink:href="//mystuff.org/mydmd001.xml"/>
     </mets:dmdSec>
     <mets:dmdSec ID="dmd002">
        <mets:mdWrap MDTYPE="DC">
          <mets:xmlData>
             <dc:record >
                 <dc:title>buying tomatoes and bananas</dc:title>
             </dc:record>
          </mets:xmlData>
        </mets:mdWrap>
      </mets:dmdSec>
      <mets:amdSec ID="amd001">
          <mets:rightsMD ID="amd001rights001">
              <mets:mdRef
                 xlink:href="https://mystuff.org/myrights001.xml"/>
           </mets:rightsMD>
       </mets:amdSec>
       <mets:structMap>
           <mets:div/>
       </mets:structMap>
</mets:mets>
```

4.5.3 Case 3: One-Sequence

In this case, one Editorial Entity is realised by a sequence of AV-media files which is a MXF-OP1A. To be noticed in the example below:

- the importance of attribute `ORDER` in the `StructMap` section
- each AV-media file points to a specific technical metadata section

```
<mets>
   <dmdSec ID="d001">
      <mdWrap MDTYPE="DC">
         <xmlData>
            <dc:identifier>urn:pprime:rai#014</dc:identifier>
         </xmlData>
      </mdWrap>
   </dmdSec>
   <amdSec ID="a001">
      <techMD ID="t001">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <techMD ID="t002">
         <mdWrap MDTYPE="OTHER">
```

```
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <rightsMD ID="r001">
         <mdRef LOCTYPE="URL" MDTYPE="OTHER"
         xlink:href="pprai014.owl"/>
      </rightsMD>
   </amdSec>
   <fileSec>
      <fileGrp>
         <file ID="f001" ADMID="t001">
            <FLocat LOCTYPE="URL"
            xlink:href="https://fileportal.rai.it/RAI_014.d10.mxf"/>
         </file>
         <file ID="f002" ADMID="t002">
            <FLocat LOCTYPE="URL"
            xlink:href="https://fileportal.rai.it/RAI_014b.d10.mxf"/>
         </file>
      </fileGrp>
   </fileSec>
   <structMap  LABEL="PPRIME-AV" TYPE="One-Sequence">
      <div TYPE="Sequence">
         <div TYPE="AudioVisual" ORDER="1"><fptr FILEID="f001"/></div>
         <div TYPE="AudioVisual" ORDER="2"><fptr FILEID="f002"/></div>
      </div>
   </structMap>
</mets>
```

4.5.4 Case 4: One-Atoms

In this case, the Editorial Entity is realised by a group of atoms, which must be played at the same time.

For each AV-media file there will be the need to point to a specific technical metadata section

```
<mets>
  <dmdSec ID="d001">
      <mdWrap MDTYPE="DC">
         <xmlData>
```

```
            <dc:identifier>urn:pprime:rai#014</dc:identifier>
         </xmlData>
      </mdWrap>
   </dmdSec>
   <amdSec ID="a001">
      <techMD ID="t001">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <techMD ID="t002">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <techMD ID="t003">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <rightsMD ID="r001">
         <mdRef LOCTYPE="URL" MDTYPE="OTHER"
           xlink:href="pprai014.owl"/>
      </rightsMD>
   </amdSec>
   <fileSec>
      <fileGrp>
         <file ID="f001" ADMID="t001">
            <FLocat LOCTYPE="URL"
            xlink:href="https://fileportal.rai.it/RAI_014.d10.m2v"/>
         </file>
         <file ID="f002" ADMID="t002">
            <FLocat LOCTYPE="URL"
            xlink:href="https://fileportal.rai.it/RAI_014.ch0.wav"/>
         </file>
         <file ID="f003" ADMID="t003">
            <FLocat LOCTYPE="URL"
            xlink:href="https://fileportal.rai.itRAI_014.ch1.wav"/>
         </file>
      </fileGrp>
```

```
    </fileSec>
    <structMap  LABEL="PPRIME-AV" TYPE="One-Atoms">
       <div TYPE="Atoms">
          <div TYPE="Video" LABEL="video"><fptr FILEID="f001"/></div>
          <div TYPE="Audio" LABEL="left"><fptr FILEID="f002"/></div>
          <div TYPE="Audio" LABEL="right"><fptr FILEID="f003"/></div>
       </div>
    </structMap>
</mets>
```

4.5.5 Case 5: One-Atoms-Sequence

In this case the Editorial Entity is realised by a sequence. Each element of the sequence can be a single AV file or a group of Atoms, to be played at the same time. At least one element in the sequence must be a group of Atoms, otherwise this case becomes analogous to the One-Sequence case.

```
<mets>
  <dmdSec ID="d001">
      <mdWrap MDTYPE="DC">
         <xmlData>
            <dc:identifier>urn:pprime:rai#014</dc:identifier>
         </xmlData>
      </mdWrap>
   </dmdSec>
   <amdSec ID="a001">
      <techMD ID="t001">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <techMD ID="t002">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <techMD ID="t003">
```

```
          <mdWrap MDTYPE="OTHER">
             <xmlData>
                <mytechnical ></mytechnical>
             </xmlData>
          </mdWrap>
       </techMD>
       <techMD ID="t004">
          <mdWrap MDTYPE="OTHER">
             <xmlData>
                <mytechnical ></mytechnical>
             </xmlData>
          </mdWrap>
       </techMD>
       <techMD ID="t005">
          <mdWrap MDTYPE="OTHER">
             <xmlData>
                <mytechnical ></mytechnical>
             </xmlData>
          </mdWrap>
       </techMD>
       <techMD ID="t006">
          <mdWrap MDTYPE="OTHER">
             <xmlData>
                <mytechnical ></mytechnical>
             </xmlData>
          </mdWrap>
       </techMD>
       <rightsMD ID="r001">
          <mdRef LOCTYPE="URL" MDTYPE="OTHER"
          xlink:href="pprai014.owl"/>
       </rightsMD>
    </amdSec>
    <fileSec>
       <fileGrp>
          <file ID="f001" ADMID="t001">
             <FLocat LOCTYPE="URL"
             xlink:href="https://fileportal.rai.it/RAI_014.d10.m2v"/>
          </file>
          <file ID="f002" ADMID="t002">
             <FLocat LOCTYPE="URL"
             xlink:href="https://fileportal.rai.it/RAI_014.ch0.wav"/>
          </file>
          <file ID="f003" ADMID="t003">
             <FLocat LOCTYPE="URL"
             xlink:href="https://fileportal.rai.it/RAI_014.ch1.wav"/>
          </file>
          <file ID="f004" ADMID="t004">
             <FLocat LOCTYPE="URL"
             xlink:href="https://fileportal.rai.it/RAI_014.d10.m2v"/>
          </file>
          <file ID="f005" ADMID="t005">
             <FLocat LOCTYPE="URL"
             xlink:href="https://fileportal.rai.it/RAI_014.ch0.wav"/>
          </file>
          <file ID="f006" ADMID="t006">
             <FLocat LOCTYPE="URL"
             xlink:href="https://fileportal.rai.it/RAI_014.ch1.wav"/>
          </file>

       </fileGrp>
    </fileSec>
    <structMap  LABEL="PPRIME-AV" TYPE="One-Atoms-Sequence">
       <div TYPE="Sequence">
         <div TYPE="Atoms" ORDER="1">
           <div TYPE="Video" LABEL="video"><fptr FILEID="f001"/></div>
           <div TYPE="Audio" LABEL="left"><fptr FILEID="f002"/></div>
           <div TYPE="Audio" LABEL="right"><fptr FILEID="f003"/></div>
         </div>
         <div TYPE="Atoms" ORDER="2">
           <div TYPE="Video" LABEL="video"><fptr FILEID="f004"/></div>
           <div TYPE="Audio" LABEL="left"><fptr FILEID="f005"/></div>
           <div TYPE="Audio" LABEL="right"><fptr FILEID="f006"/></div>
         </div>
       </div>
    </structMap>
 </mets>
```

4.5.6 Case 6: Multi-one

In this case the submitted AV material consists of multiple different copies (for different uses) realising the same Editorial Entity.

In the example one is the Master and one is for exploitation. Normally the exploitation/access/browsing quality copies might be created after the submission by the (extended-)preservation system. The master copy is identified in the structural map section with the `div` attribute `LABEL="Master"`.

```
<mets>
  <dmdSec ID="d001">
      <mdWrap MDTYPE="DC">
         <xmlData>
            <dc:identifier>urn:pprime:rai#014</dc:identifier>
         </xmlData>
      </mdWrap>
   </dmdSec>
   <amdSec ID="a001">
      <techMD ID="t001">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <techMD ID="t002">
         <mdWrap MDTYPE="OTHER">
            <xmlData>
               <mytechnical ></mytechnical>
            </xmlData>
         </mdWrap>
      </techMD>
      <rightsMD ID="r001">
         <mdRef LOCTYPE="URL" MDTYPE="OTHER"
         xlink:href="pprai014.owl"/>
      </rightsMD>
   </amdSec>
   <fileSec>
      <fileGrp>
         <file ID="f001" ADMID="t001">
            <FLocat LOCTYPE="URL"
            xlink:href="https://fileportal.rai.it/RAI_014.d10.mxf"/>
         </file>
         <file ID="f002" ADMID="t002">
            <FLocat LOCTYPE="URL"
            xlink:href="https://fileportal.rai.it/RAI_014.ogv"/>
         </file>
      </fileGrp>
   </fileSec>
   <structMap  LABEL="PPRIME-AV" TYPE="Multi">
      <div TYPE="MultiCopy">
         <div TYPE="AudioVisual" LABEL="Master">
             <fptr FILEID="f001"/>
         </div>
```

```
          <div TYPE="AudioVisual">
             <fptr FILEID="f002"/>
          </div>
       </div>
    </structMap>
 </mets>
```

5 Reference Architecture

In PrestoPRIME, digital preservation is extended to take into account digital contents provided by broadcasters, which are bringing something new in the field of digital preservation: high quality videos requiring innovative solutions for storing the items, monitoring their status and managing the obsolescence of encodings and file formats. PrestoPRIME is facing the challenge of dealing with compression issues, format changes and the big file sizes of the high quality or master copies.

In order to address the problem, a first analysis of the *Preservation Platform Architecture* was carried out in 2009 (the first year of the project), focusing the attention on the integration of the software modules that the project Work Packages provide. This first attempt is described in Section 5.1. It was soon evident that an *integration infrastructure* is too general for the complex digital preservation issues and after a deep survey on the state of the art, it was decided to adopt the OAIS [8] standard, merging the analysis already done with it and taking into account the best practices of preservation system leaders such as Rosetta [4].

Furthermore, a novel approach to the design of the system was followed: define the needed *abstract components* and get the architecture as a result. These *abstract components* are taken from the Object Oriented Design and are described as *design patterns*, the building blocks of our preservation platform. This novel approach is described in Section 5.2.

PrestoPRIME is developing tools and techniques to support the transitions that AV archives are facing today, moving to file-based content and IT systems. Technologies developed by PrestoPRIME include tools for calculating the long-term Total Cost of Ownership (TCO) of these systems, planning and optimising file format migration, assessing and comparing what storage technologies or services to use, managing the risks involved, selecting suitable preservation strategies (e.g. file format migration, emulation, multivalent), and automating content quality control, e.g. defect detection and analysis.

Recognizing that automation is the key to lowering costs (and risks), PrestoPRIME is supporting digital preservation infrastructure components that have the ability to execute preservation policies and processes and make use of managed storage as a service that conforms to defined Service Level Agreements and QoS.

5.1 Towards the Preservation Platform

Deliverable 5.1.1 [1] collected the scenarios and user requirements that PrestoPRIME has to fulfill, and consequently the Preservation Platform has to implement (see table 10). Concerning the *roles* to be supported, we can recognize the following:

- the **Producer**, responsible for providing the digital contents, setting the properties and rights. From [8] we can recall:

> "The role played by those persons, or client systems, who provide the information to be preserved. This can include other OAISs or internal OAIS persons or systems."

- the **Consumer**, the actor that applies for searches, able to browse and edit results (Information Packages). From [8] we recall:

> "The role played by those persons, or client systems, who interact with OAIS services to find preserved information of interest and to access that information in detail. This can include other OAISs, as well as internal OAIS persons or systems."

- the **Manager**, responsible for setting policies and administering the workflows. From [8] we recall:

> "The role played by those who set overall OAIS policy as one component in a broader policy domain, for example as part of a larger organization."

These *roles* can be interpreted by different *actors*. From the point of view of the overall system, our stakeholders are:

- Broadcasters
- Digital libraries
- Service Providers
- Technologists
- Researchers
- Users (non-qualified user, providing his own generated contents)

Taking into account these actors and scenarios, during the first phase of the project (2009), available standards and technologies were analised in order to set up an open and flexible digital preservation platform. The State of the Art Section describes similar projects and some well established solutions.

The overall approach adopted in PrestoPRIME for software design is the Model Driven (Model Driven Architecture MDA) [73] developed by the Object Management Group (OMG) [74], which is also responsible for the UML [2] language. It has been proven that it's profitable to take into account the best practices and already available solutions for defining the software architecture, making use of the well established design patterns addressing similar needs. Having the MDA in mind, during the first year of the PrestoPRIME project, the first approach taken was the classic one for software integration: the enterprise service bus (ESB [75]), with a Service Oriented Architecture (SOA [76]), where the protocols chosen for connecting the different and distributed software components are SOAP and REST as follows:

- Simple Object Access Protocol (SOAP [77]) to be used each time a specific schema is required. SOAP is restrictive enough on semantics. Services are published as *Web Services* by means of Web Services Description Language (WSDL [78]) interfaces.
- REpresentational State Transfer (REST [79]) services to be used each time the exchange XML documents is performed with no constraints on their schemas.

The reason for adopting REST and not only SOAP is that sometimes developers make use of SOAP with RPC [2] and DOM [3] without any control of the data type transferred. In those cases it is preferable and more flexible to adopt RESTful services in order to leave schemas to be redefined and the exchanged XML open to further changes and improvements. RESTful web services are becoming increasingly widespread, for example in the linked open data community that could be relevant for metadata enrichment. WSDL 2.0 provides support for describing such services.

Figure 20 shows an overview of the PrestoPRIME Platform components with the enterprise service bus approach. As the reader can easily recognize, the protocols and interfaces have been pointed out in the diagram.

[2] Remote Procedure Call
[3] Document Object Model

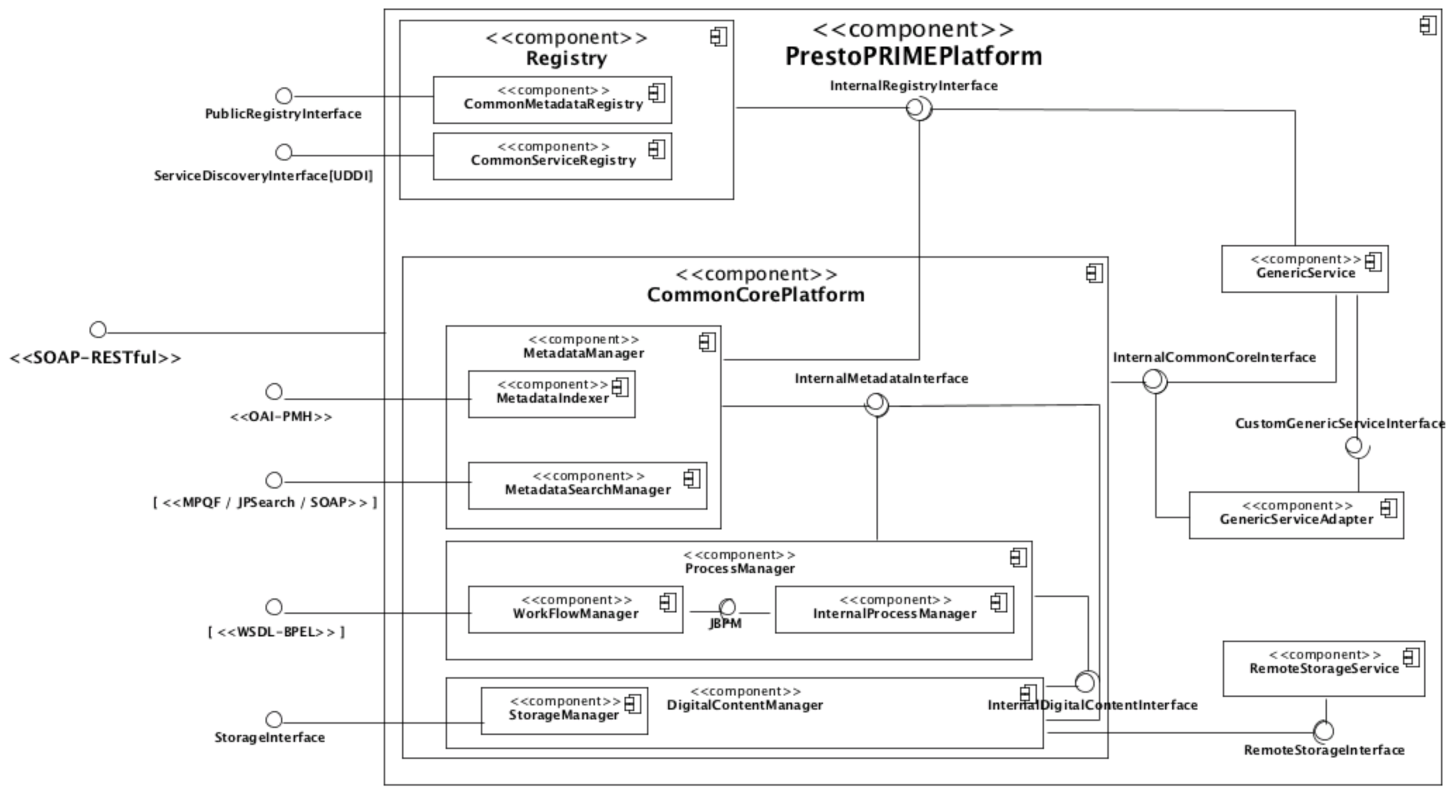

Figure 20: *First version of the PrestoPRIME Platform Components*

The component diagram displays a main module *CommonCorePlatform*, in which the main functions are implemented. The overall Platform comprises a Registry module and makes use of the Adapter pattern, as the dynamic plug-in facility to the system.

As already said, Figure 20 is designed starting from the protocols and interfaces in order to be able to integrate other software components. From the top left there are the UDDI [80] interface for the Registries, the SOAP-RESTful interface as the main one for accessing the *CommonCorePlatform*, the metadata and content based (CBIR[4]) protocols for querying the catalogue with complex structures and BPEL [81] for accessing (potentially privately) the work flow module and finally the Storage interface for accessing the low level storage devices which are hidden inside the overall platform.

Adapters are implemented for any component out of the scope of the framework. The adapters talk SOAP to the software modules and to the framework as well. This mechanism is useful for connecting new software components, especially at a later time of development, enabling a hot plug in of new tools.

Looking at the top left corner of Figure 20, a common registry is depicted. According to the analysis on needed registries for a preservation platform, which is still under development, this registry component is a placeholder for a service looking up resources as well as distributed services. This registry may also expose the Universal Description, Discovery, and Integration (UDDI [80]) service for finding out the right (and working) interface, enabling a load balancing of the remote services.

The analysis which led to this first architecture was mainly focused on the *integration* technical issues, without much attention to the OAIS requirements. Soon and after a deeply analysis of the OAIS [8] framework, the project partners decided to adopt it as reference for PrestoPRIME and the design had to be rewritten. As a starting point, already existing projects implementing OAIS were taken into account, with the goal to find available solutions ready to be used, or designs to follow. Most of these related projects are reported in Section 2. One of the most relevant projects was CASPAR, described in Section 2.1, which tried to map the OAIS functional blocks into software components. Unfortunately, as shown in Figure 2 the mapping was quite straightforward, sometime one-to-one without much more design added. Looking at the software implementation, it turned out that it wasn't so easy to start from it for the purposes of PrestoPRIME and it was decided to start designing a new platform.

Recently a further project emerged, which in some way is very close to PrestoPRIME: the Preserving Digital Public Television Project (PDPTV [61]) of the National Digital Information Infrastructure and Preservation Program, which has delivered an important report [63] on software architecture for preserving digital contents in the broadcasting environments. Figure 21 shows the mapping of OAIS functional entities to preservation repository software components and points out the underlying software used, DSpace [17] (described in Section 2.6). Some of the the DSpace components, such as the *Importer* and the *OAI-PMH Data Provider* are useful in our context, ready to be used and which will be taken

[4]Content Based Information Retrieval

into account in the actual implementation [3]. Even if DSpace is a reasonable choice,

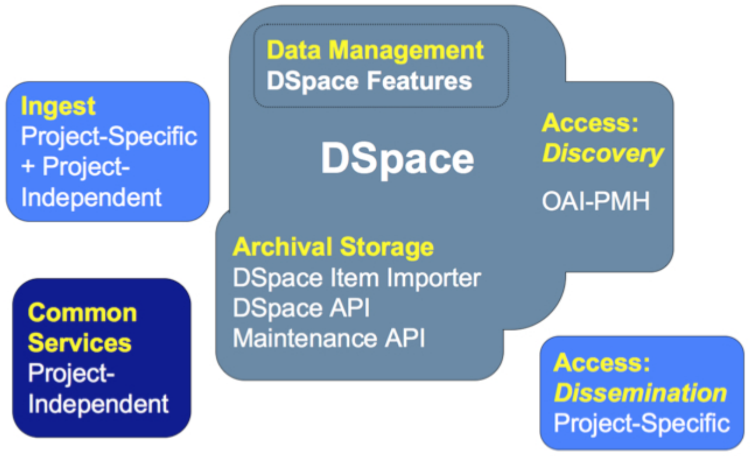

Figure 21: *The OAIS Functional Entity to Preservation Repository Software Component Mapping* [63]. *(*OAIS Functional Entity names appear in yellow)

another approach was adopted for the design of the PrestoPRIME Preservation Platform, with a more abstract purpose: try to identify the *abstract* software components needed to implement the PrestoPRIME requirements and especially try to design the Platform Independent Model. The Specific Model (PSM) and the actual implementation will follow accordingly. There will be two software specific implementations, described in D5.2.2 [3] and D5.3 [5, 6].

Report [63] helped to check that the main choices made so far for PrestoPRIME architecture are aligned (if not the same) with PDPTV. As thoroughly described in [64], a preservation system dealing with broadcasters should start from the OAIS specifications and take into account the TRAC [9]. In the following Section 5.2 a detailed OAIS mapping is presented and in Section 5.3 the SLA approach is described, which tries to meet the TRAC guidelines. Also the choice of having a standard wrapper such as METS for the SIP is aligned with PDPTV (see Section 4 and Section 3.2.4). Concerning the Metadata and Content inside a SIP, it is a matter of requirements to be met. In PrestoPRIME we have to handle high quality video files and a *content wrapper* such as MXF is again an aligned choice. Concerning Metadata, in order to be flexible enough and open to future extensions, metadata structure are not to be restrictive. Possible metadata formats are listed in Deliverable 2.2.2 [52] (e.g. MPEG-7 DAVP, MPEG-7 ECM SCAIE, P_Meta, PrestoSpace EDOB, DC/DCterms, EBUcore, PBCore) and will be defined in Deliverable D2.1.3 [70]. Also the work performed on the definition of a *Rights Ontology* able to map broadcasting contracts (Work Package 4 Task 4) has proven that simpler structures such as METSRights or MPEG-21 REL are not able to handle the complexity and the several self-relationships between the many *graphs* describing a contract and consequently its exploitation rights. Once again, it's woth putting a stress on the importance of keeping an abstract level of description in order to be able to adapt/implement the PrestoPRIME platform in a wide range of digital contexts.

The main abstract design is presented Section 5.2, while Section 5.3 describes the SLA based management and the needed components. Conclusions and a wrap up table are reported in Section 6.

5.2 The Preservation Platform Design

During the first year of the project, the OAIS model [8] was deeply analysed framework and eventually adopted as the reference for the architecture of the PrestoPRIME Preservation Platform. Figure 22 depicts the functional entities of an Open Archival Information System (OAIS). In the following, the first analysis described in Section 5.1 is improved,

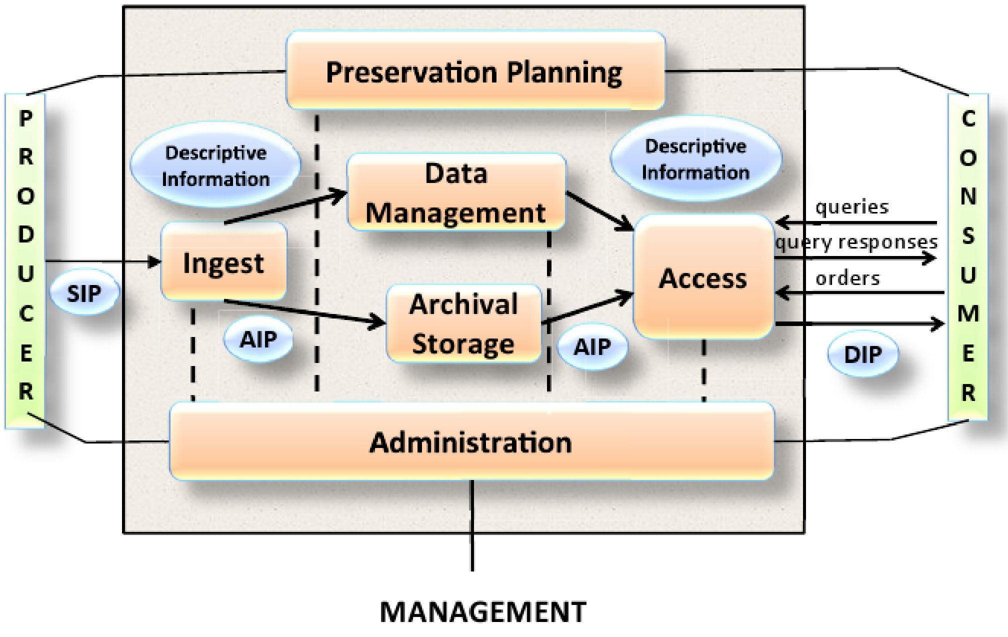

Figure 22: *The OAIS [8] functional entities*

adopting the OAIS specifications as reference and taking into account the Model Driven Architecture guidelines [73] and the best practices implemented in the Rosetta [4] commercial system, that is one of the current market leaders.

In order to achieve the highest flexibility and to enable an easy plug-in of new software components into the platform, the ESB [75] (Enterprise Service Bus) architecture is taken, as described in previous Section 5.1, where services are provided by SOAP each time a specific schema is required leaving REST to manage the protocol each time it is necessary to exchange plain XML documents.

If a pre-existing digital preservation system follows the OAIS guidelines, the PrestoPRIME solution can be easily integrated with it and can help to manage some specific issues not already covered. Moreover, the PrestoPRIME platform can be used to migrate an old archive to a new one, based upon standards and best practices. Finally, some isolated tool can be extracted and plugged into an existing system.

The core functionalities available in the PrestoPRIME Preservation Platform will be released by the Consortium as software libraries under an open license (see Section 5.4). At the same time a commercial implementation will be available, provided by the ExLibris Company [7].

The novel approach we are providing in this document is the MDA design of the *software components* that implement the *functional entities* shown in Figure 22.

Assuming the Object Oriented Design (OOD) and the Object Oriented Programming (OOP) within the Preservation Platform, we can make use of the well established Design Patterns [82, 83, 84] already solving specific software problems, leading up to an overall MDA approach for the PrestoPRIME software development.

"*One way to avoid the act of design is to reuse existing designs*" [85]: the approach initially proposed by Beck and Johnson has been fully embraced. The overall analysis and design of the PrestoPRIME Preservation Platform reference architecture maps the six OAIS functional entities into software components making use of available design patterns. *Generative patterns* [86, 87] are used for addressing the specific requirements of each of the following functional entities:

- ingest
- access
- administration
- preservation planning
- data management
- archival storage

The design patterns solving the problems are recognized and brought together in order to build up the software component diagram of the functional entities.

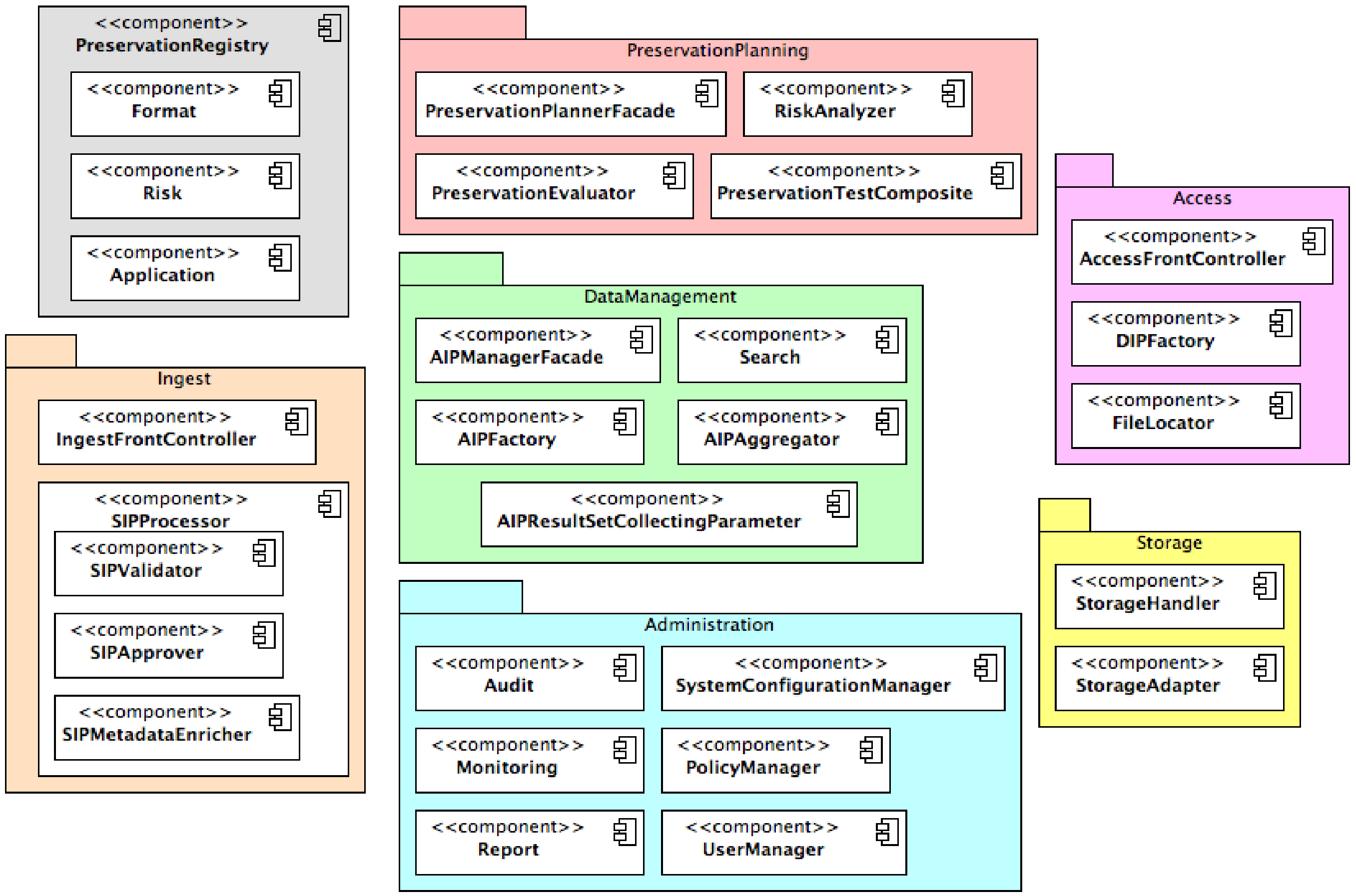
<<component>> PreservationRegistry
<<component>> Format
<<component>> Risk
<<component>> Application
PreservationPlanning
<<component>> PreservationPlannerFacade
<<component>> RiskAnalyzer
<<component>> PreservationEvaluator
<<component>> PreservationTestComposite
Access
<<component>> AccessFrontController
<<component>> DIPFactory
<<component>> FileLocator
DataManagement
<<component>> AIPManagerFacade
<<component>> Search
<<component>> AIPFactory
<<component>> AIPAggregator
<<component>> AIPResultSetCollectingParameter
Ingest
<<component>> IngestFrontController
<<component>> SIPProcessor
<<component>> SIPValidator
<<component>> SIPApprover
<<component>> SIPMetadataEnricher
Storage
<<component>> StorageHandler
<<component>> StorageAdapter
Administration
<<component>> Audit
<<component>> SystemConfigurationManager
<<component>> Monitoring
<<component>> PolicyManager
<<component>> Report
<<component>> UserManager

Figure 23: *PrestoPRIME Preservation Platform Component Diagram*

Figure 23 shows a simplified overview of the software design (component diagram), pointing out the conceptualised components for each of the six OAIS functional entitites [5].

A software component not planned in the OAIS model has been added to the figure (top left): the *Preservation Registry* that the platform has to contact during most of the processes, such as ingestion (*SIPProcessing*) and *RiskAnalysis* (Preservation Planning). What has been described here is a candidate *Preservation Registry* designed from the experience of Rosetta [4] and PRONOM [38]. A further and more detailed analysis on the *registries* needed in a preservation system [6] is planned for the next year. For the Architecture Design the discussion is limited to the Registry represented in Figure 24. Figure 24 shows how the components of the Preservation Registry are related in a class diagram.

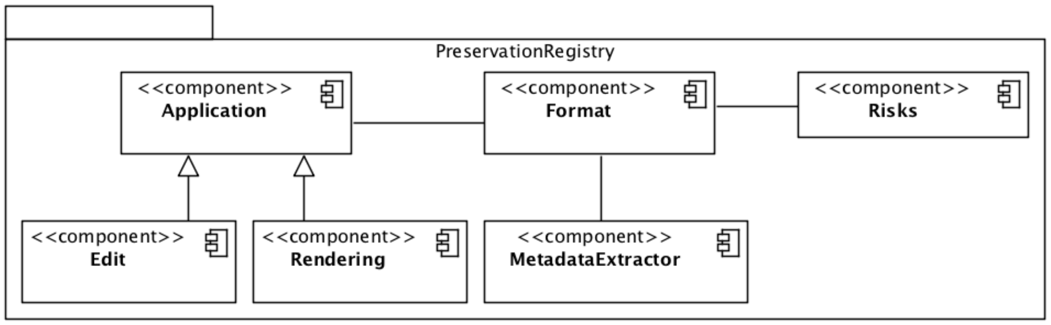

Figure 24: *Preservation Registry class diagram*

The Preservation Registry is made up of three main components:

- the **Format**, which is responsible for identifing the format and type of the digital content. Associated to the Format there's the MetadataExtractors, that are available for a specific Format and type (for example for PDF Format and Type version 2);
- the **Risk**, which represents the risk associated to a specific Format (and type);
- the **Application**, which can be split into Editing and Rendering Applications, for example coding video contents or playing a movie.

These kinds of information are stored into a *Preservation Registry* that should be distributed. In the PrestoPRIME context, there is a Registry for each Archive and a centralised one (gathering information from all the Archives) it is planned in order to provide a federated service for automatic preservation.

[5]The reader should take care that the OAIS activities and functions in each entity are not mentioned because the diagram is focused on the software structure, not behavior (more precisely the Platform Independent Model [73] design)

[6]see deliverable D5.2.3- M36

Some software components are not displayed in Figure 23 (for better readability), such as the processing layer named *Workflow Module*, a cross software component used by the others, responsible for managing the workflow that the several processes need during the runtime phases, such as ingestion and preservation.

For each of the six OAIS functional entities planned in Figure 22 and depicted in Figure 23, a more detailed description (with the associated component diagrams) follows.

5.2.1 Ingest

During the Ingest phase several steps have to be taken. First of all, a ***FrontController*** [88] design pattern is needed in order to get the requests and dispatch them to the associated component. The *IngestFrontController* is the first component contacted by the

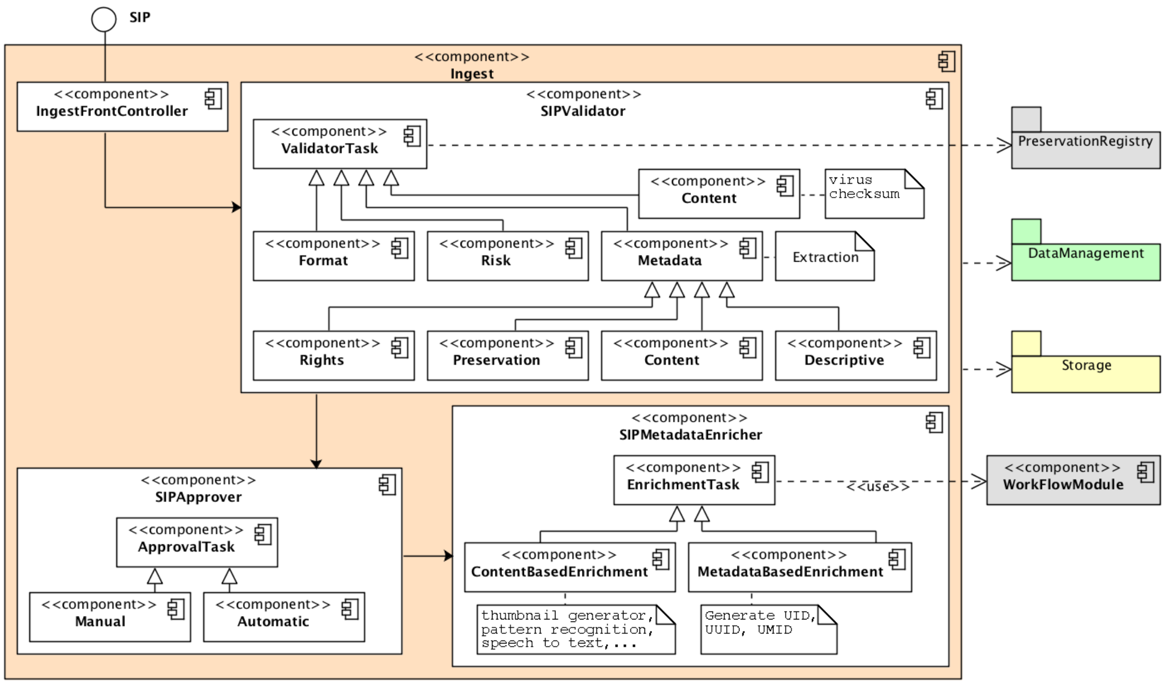
SIP
<<component>> Ingest
<<component>> IngestFrontController
<<component>> SIPValidator
<<component>> ValidatorTask
<<component>> Content
virus checksum
<<component>> Format
<<component>> Risk
<<component>> Metadata
Extraction
<<component>> Rights
<<component>> Preservation
<<component>> Content
<<component>> Descriptive
PreservationRegistry
DataManagement
Storage
<<component>> SIPMetadataEnricher
<<component>> EnrichmentTask
<<use>>
<<component>> WorkFlowModule
<<component>> SIPApprover
<<component>> ApprovalTask
<<component>> Manual
<<component>> Automatic
<<component>> ContentBasedEnrichment
<<component>> MetadataBasedEnrichment
thumbnail generator, pattern recognition, speech to text, ...
Generate UID, UUID, UMID

Figure 25: *Ingest Component Diagram*

Producer. Then three main components can be recognised in Figure 25:

- **SIPValidator**, which is responsible for the overall validation of the SIP submitted by the *Producer*. This process requires the validation of the Formats (such as the check of wellformedness and validity), the Metadata, the Content (such as virus checks and checksums), and Rights validation. Concerning the Metadata, a further inheritance is depicted, pointing out Metadata for Preservation, Rights, Content and Descriptive. For each one a specific format is defined and accepted in the Validation process, as already described in the SIP Section of this document.

- **SIPApprover**, which is responsible for approving the several Tasks performed in the Validation process, which can be Automatic (most of them) as well as Manual (requiring human intervention).
- **SIPMetadataEnricher**, which is responsible for Enriching the Metadata describing the digital item submitted. These could be categorised into two main classes: Metadata extracted from the content, such as speech to text and visual descriptors and Metadata added to the item such as the UUID [89] or UMID [90] and format changes or aggregations.

Once the SIP has been analysed and validated, it can be stored into the Archive and the Preservation Process can take place.

There are already several tools providing validation and metadata extraction features, such as JHOVE [59], DROID [40] and NZNL Metadata Extractor [91] and they will be taken into account during the Preservation Platform implementation phase.

As Figure 25 shows, the main interface exposed by the Ingest component is the SIP, where other components can ask to submit SIPs. On the right side the main dependencies are shown. The Validation Task makes use of the Preservation Registry, the Enrichment Task makes use of the Work Flow Module for processing the Content and Metadata Enrichment tasks. The overall Ingest makes use of the DataManagement (the first one contacted) and Storage components that are described in the following.

Used Design Patterns

FrontController

5.2.2 DataManagement

The ingested SIPs are mapped onto AIPs. Within the DataManagement component, a ***Façade*** [83] design pattern is needed, as uniform and single interface for accessing AIPs. The *AIPManagerFaçade* is the façade component, as displayed in Figure 26.

The AIPMangerFaçade contacts a *Search* component which is responsible for searching and aggregating AIPs. In order to get the AIPs, it makes use of a ***Factory*** [83] for creating new AIPs as well as accessing preexisting ones, leading up to an AIPFactory implementation. The AIPFactory makes use of the ***Entity Access Object*** (EAO) [92, 93] design pattern for accessing the actual AIP, that adopts the AIP entities instead of simple Objects in the previous Data Access Object (DAO) [94] pattern.

So far, this discussion has been focused on the creation and access of AIPs. The Data Management package is also responsible for supporting searches and aggregate results that are realised by a *Search* and an *Aggregator* components. The former is accessed by the Façade and performs the queries on AIPs. The latter provides the aggregations of

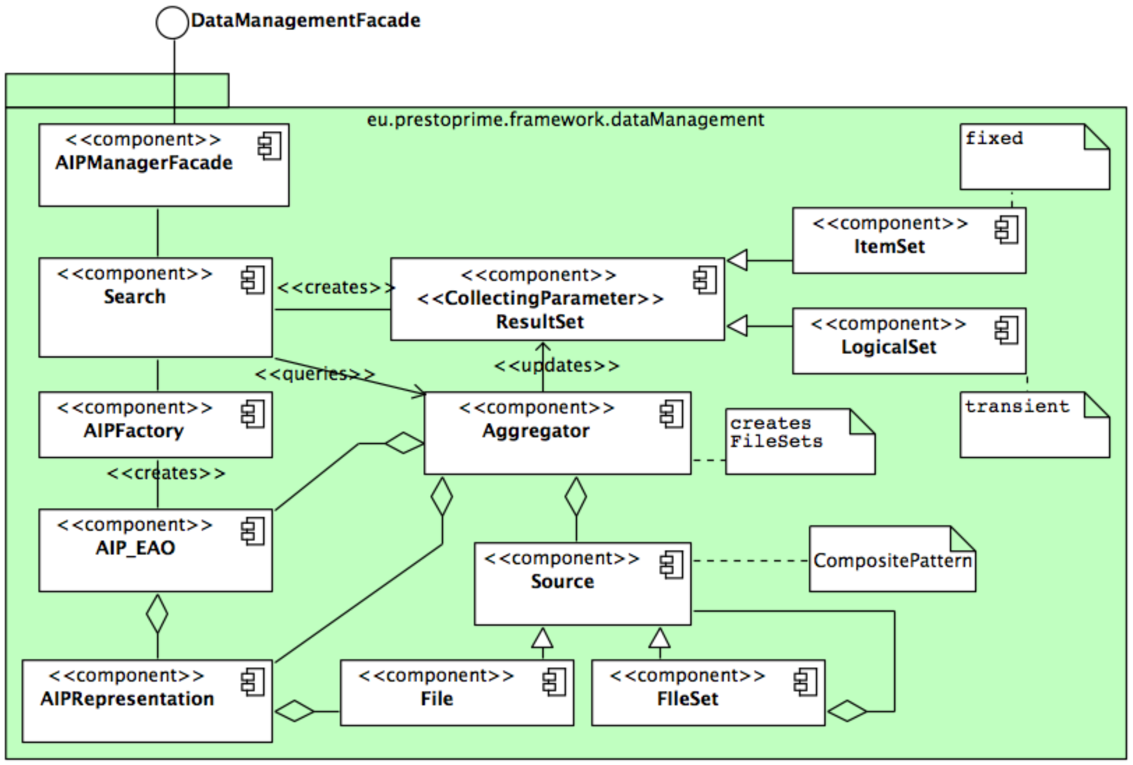

Figure 26: *Data Management Component Diagram*

several items matching the search criteria. In order to have a common result in a single entity, the ***CollectingParameter*** [83, 95] design pattern is introduced. In this pattern an *ResultSet* object is passed through the Search and Aggregator components, back to the Façade. The ResultSet is extended by *ItemSet*, which describes fixed and real items, and *LogicalSet*, which describes transient items such as aggregations of the same format type of items. The *Aggregator* component will be used in most of the internal processes executed for the preservation purposes such as evaluation of the risks associated to a specific format, or type, or the availability of a rendering application associated to a given media content. In order to deal with the aggregated sources as single items as well as item sets, a ***Composite Pattern*** [83] is adopted: the Aggregator aggregates Source which is inherited by File (the leaf) and FileSet (the aggregation of Source). The leaf File is associated and aggregated by the *AIPRepresentation* which is aggregated by the Aggregator.

Used Design Patterns

Façade, Factory (AbstractFactory), Entity Access Object, Composite, Collecting Parameter

5.2.3 PreservationPlanning

The Preservation Planning package should expose a common interface to the other internal software components: a ***Façade*** pattern has been chosen. It is implemented by the *PreservationPlannerFaçade* component as described in Figure 23. In Figure 27 it has been omitted for better readability.

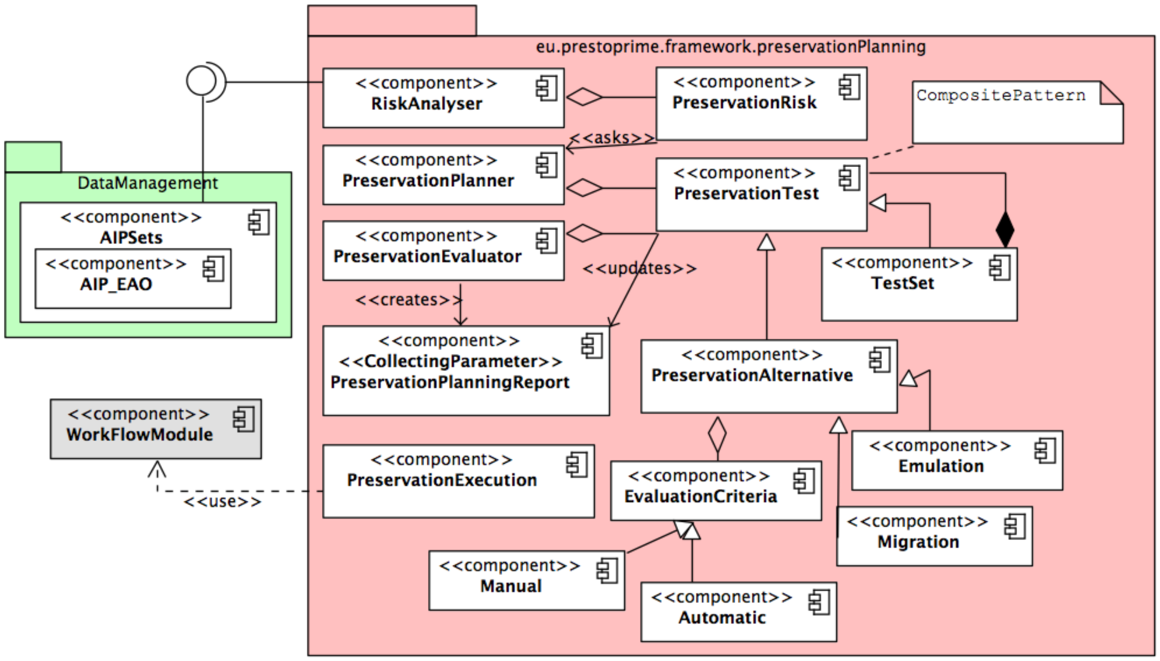
eu.prestoprime.framework.preservationPlanning
<<component>> RiskAnalyser
<<component>> PreservationRisk
CompositePattern
<<asks>>
<<component>> PreservationPlanner
<<component>> PreservationTest
DataManagement
<<component>> AIPSets
<<component>> AIP_EAO
<<component>> PreservationEvaluator
<<updates>>
<<component>> TestSet
<<creates>>
<<component>> <<CollectingParameter>> PreservationPlanningReport
<<component>> PreservationAlternative
<<component>> WorkFlowModule
<<component>> Emulation
<<component>> PreservationExecution
<<component>> EvaluationCriteria
<<use>>
<<component>> Migration
<<component>> Manual
<<component>> Automatic

Figure 27: *Preservation Planning Component Diagram*

The package is responsible for evaluating risks (charged to the *RiskAnalyser* components), perform preservation simulations as well as the several preservation tasks required by the archive. The Risk Analyser makes use of the DataManagement package, as shown by the exposed half circle attached to the lollipop of the DataManagement in Figure 27), where only the used components are displayed, the AIPSets and the Entity Objects. In particular, the AIPSets are the items (in Figure 26 are pointed out more “generic” *ItemSet*) requested to the DataManagement on which the preservation simulations are performed.

Risk Analyser aggregates the *PreservationRisk*, which asks for a preservation plan to the *PreservationPlanner*. The latter aggregates the *PreservationTest*. In order to have a common rule for managing single tests as well as aggregations of tests, a ***Composite*** pattern is introduced, where an abstract *PreservationTest* is inherited by the leaf *PreservationAlternative* and the aggregator *TestSet*.

The PreservationTest is aggregated by the *PreservationEvaluator*, which creates the Result objects making use of the ***CollectingParameter*** pattern, implemented by the *PreservationPlanningReport*. It is passed to the PreservationTest abstraction, which updates it each time a leaf test is performed. For each Test there are alternatives to take into account. The *PreservationAlternative* is inherited by the *Emulation* and the *Migration* which are the two possible implementations and aggregates the *EvaluationCriteria*. The latter is inherited by the *Manual* and the *Automatic* criteria for the evaluation of the preservation alternatives.

Ideally the best preservation platform should avoid the intervention of humans and should make use of the *Automatic* component. In those cases in which the user should be involved in the evaluation, the Manual component will interact with the appropriate UI (User Interface). As a result of the preservation planning the *PreservationPlanningReport* is provided.

The software component displayed in Figure 27 linked to the others is the *PreservationExecution*, which makes use of the *WorkFlow* module. The Preservation Execution is a complex component and will expose hooks for custom implementations and enabling a plug-in feature of different executors.

Used Design Patterns

Façade, Composite, Collecting Parameter

5.2.4 Storage

In the Storage package there is the need to decouple the storage devices from the software components running in the OAIS Archive. A *StorageHandler* is responsible for providing the storage functionalities such as the read and write methods, talking to an ***Adapter*** [83] implemented by the *StorageAdapter* component, that maps the generalised methods to the specific attached storage devices.

Figure 28 shows a “dummy” component named *AnyStorageDevice*, which extends the StorageAdapter, that represents a specific storage device such as for example a Local File System or Network Attached Storage. The Local File System can be considered as the default storage device implemented which should be always available to the platform. Tapes will be covered as well and generally, for each new device to be managed, a new Adapter should be provided. The used approach is similar to the *driver* design.

The overall Storage package exposes a ***Façade*** pattern implemented by the lollipop symbol in Figure 28. The interface is implemented by the StorageHandler which is open to further extensions. The *Adapter* approach enables future implementation that can be plugged-in to perform the actual storage operations.

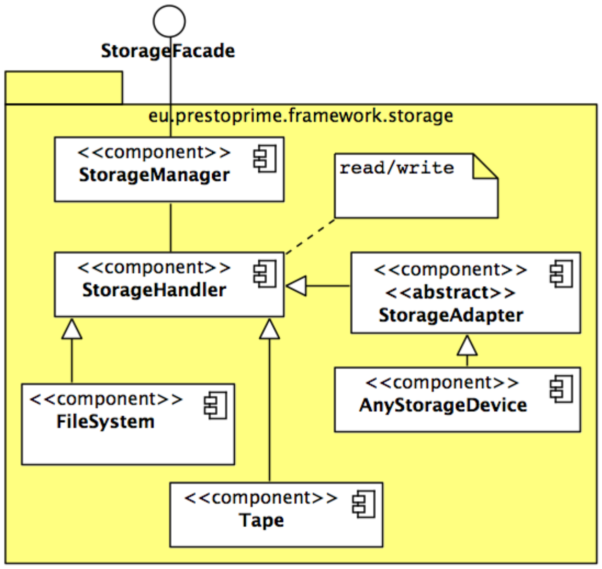
StorageFacade
eu.prestoprime.framework.storage
<<component>> StorageManager
read/write
<<component>> StorageHandler
<<component>> <<abstract>> StorageAdapter
<<component>> AnyStorageDevice
<<component>> FileSystem
<<component>> Tape

Figure 28: *Storage Component Diagram*

Used Design Patterns

Façade, Adapter

5.2.5 Access

The *Access* package is connected to the actor *Consumer* as shown in Figure 22. Whenever an actor is connected to the OAIS archive (as for example the Ingest package connected to the *Producer*) a ***FrontController*** pattern is used. The FrontController is responsible for accepting the incoming requests and for dispatching them to the right software component in the package. In the Access package, the FrontController is implemented by the *AccessFrontController* component which exposes also a WS (WebServices) interface, as displayed in Figure 29.

Again, in order to manage the creation and find of DIP packages, a ***Factory*** is needed, implemented by the *DIPFactory* component, which is associated to (and used by) the AccessFrontController.

The DIPFactory makes use of the *DataManagement* for searching the required Information Package. If it is found, according to the Consumer request, a set of tasks is executed, submitted to the *WorkFlowModule*. These tasks could be for example the translation of the Information Package to a low resolution format as well as the resize and trim of the items realising the IP.

Once the DIP is created by the factory, the consumer needs to access its contents. They are placed/handled in the Storage package and in order to locate them we need a decou-

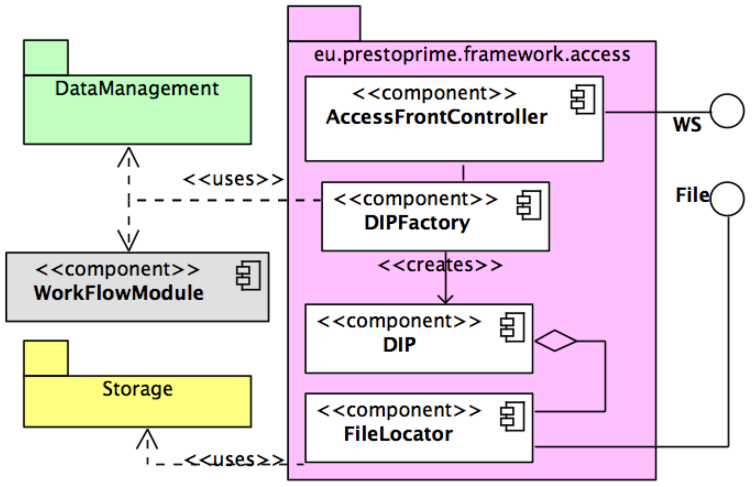

Figure 29: *Access Component Diagram*

pling service: a ***Locator*** [88] is introduced, implemented by the *FileLocator* component which locates the contents and resources associated to the AIPs.

The FileLocator component exposes also a *File* interface to the user (Consumer) which will be used for accessing the actual resources.

The Access component is also responsible for providing the user with functionalities for querying the Archive. If the user asks for a query, the AccessFrontController forwards the request to the DataManagement package where the Search component implements the different searching functionalities. The following protocols are under evaluation:

- OAI-PMH [96]
- MPEG Query Format (MPQF) (ISO/IEC 15938-12) [97]
- JPSearch3 [98]

Whilst the first one is the most widely used and implemented within the digital libraries, the last two have emerged in the latest years, enabling more complex queries. For example MPQF enables range and kNN queries and follows the Content Base Information Retrieval approach, in which the user can submit a digital content asking the system to search for similar ones. It is assumed that the OAI-PMH will be implemented as the default one by the Access component in the PrestoPRIME Preservation Platform.

Used Design Patterns

Front Controller, Factory, Locator

5.2.6 Administration

The Administration package is responsible for:

- the **system configuration**, **user profiling** and **policy settings**, implemented by the *SystemConfigurationManager*, the *UserManager* and the *PolicyManager* components respectively.
- **monitoring** the overall preservation activities, implemented by the *Monitoring* component
- providing the **audits** to the user, functionality implemented by the *Audit* component, making use of a ***CollectingParameter*** pattern, realised by the the *Report* component.

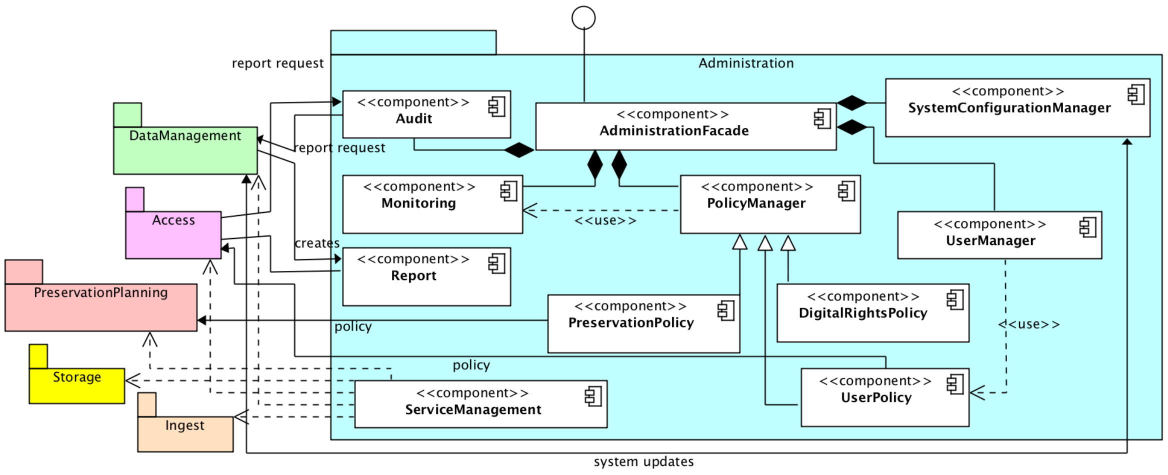

Figure 30: *Administration Component Diagram*

As shown in Figure 30, the Administration package makes use of (and thus depends on) DataManagement, Access and PreservationPlanning. The user (the actor *Administrator*) as well as other software components or systems can access the administration module by means of the *AdministrationFacade* which implements the **Façade** pattern and the lollipop interface depicted in Figure 30. The Façade component aggregates[7] the SystemConfigurationManager, the UserManager, the PolicyManager, the Monitoring and the Audit components. The SystemConfigurationManager keeps the DataManagement updated. The DataManagement creates *Report*s that as "CollectingParameter" are passed to the Audit which fills them. The UserManager is responsible for the management of the users accessing the Archive, real as well as logical ones. It makes use of the UserPolicy component which is a child of the *PolicyManager* and is aligned to the Access module.

[7]It is a strong aggregation pointed out by the black diamond that means "composition" rather than a simple association

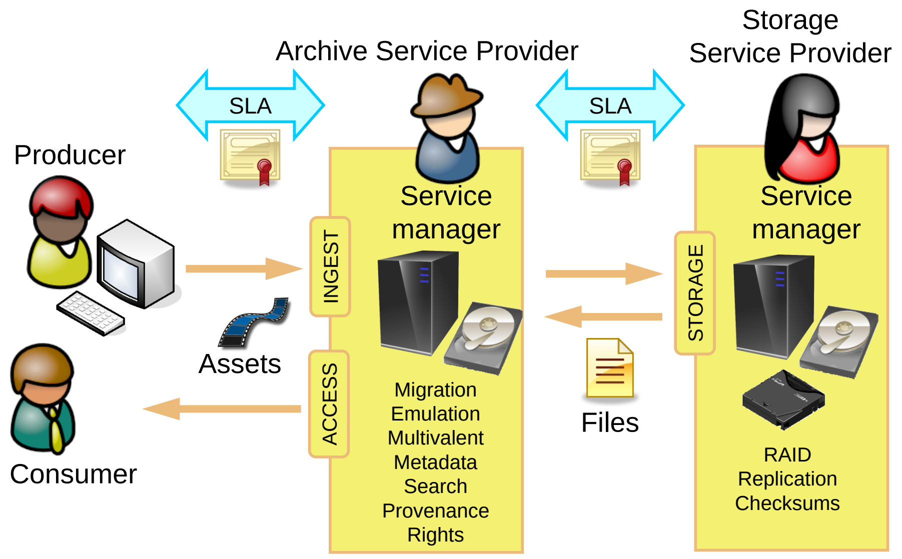

Figure 31: Preservation Supply Chain

The latter is also the superclass of the *DigitalRightsPolicy* and the *PreservationPolicy* which are some (but not all) of the available inheritances/specialisations. If new needs arise, new children can be added to the PolicyManger, implementing their management. The PolicyManager makes use of the *Monitoring* component, which is also aggregated by the AdministrationFacade. It is worth to note that the PolicyManager child dealing with the preservation, the PreservationPolicy, is strongly connected to the PreservationPlanning package. A further component shown in Figure 30 is the *SLAManagement* that will be described in Section 5.3, which interacting with the other functional blocks (depicted as dependencies arrows in the diagram), makes use of a SLA, Service and Resource Managers. According to OAIS, the administration functions include:

- soliciting and negotiating submission agreements with Producers
- maintaining configuration management of system hardware and software
- providing system engineering functions to monitor and improve archive operations

These functions accord well with those of the SLA management components described in detail in Section 5.3.4.

Used Design Patterns

5.3 SLA-based Management Framework

5.3.1 The Preservation Supply Chain

Figure 31 shows the preservation supply chain with the content producers and consumers both on the left-hand side, as customers of the archive service provider. Their relationship with the archive is described by the SLA (or SLAs) between them and the archive. On the right-hand side, the archive has an SLA with third-party suppliers that provide external storage and compute services. The archive must manage both its commitments to its customers as well as commitments to its suppliers according to the respective SLAs.

The technical approach proposed here is to treat each ICT component of the archive as a service. An individual service may provide, for example, transcoding, storage, or integrity checking. The behavior of a service can be described by a number of metrics encoded in an SLA. The types of metrics and SLAs of interest to the preservation community were described in [ID3.4.1/Section 3]. An SLA is agreed between the service provider and consumers of this service (humans or other services that may use it as part of a workflow). This allows interconnections between services to be managed in order to maintain the overall requirements for the archive system in which they are used. The archive service provider is then able to specify his own system characteristics for the services (ingest and access) it in turn provides to content producers and consumers.

5.3.2 A Flexible System Management Approach

The key objectives of the proposed approach to management are the following:

- to encode service-level commitments in a machine readable way, so they can be included in SLAs;
- to use service governance mechanisms to ensure these commitments will be met, using monitoring and management of available resources (which may themselves be services).

Figure 32 shows the important control loops that exist within a managed infrastructure. Management in this context is concerned with sending controlling signals to the ICT components (the archive services) when monitoring data indicates a need to do so. Without a management framework, the ICT operators can of course monitor information supplied by ICT components used within the archive, and take action when they consider this information indicates a need to do so. This provides a ‘humanised’ or ‘manual’ management loop between the operators and the infrastructure, indicated on the left-hand side of Figure 32.

The management framework monitors the ICT infrastructure and uses a common Web service management interface to manage the dynamic composition of services and un-

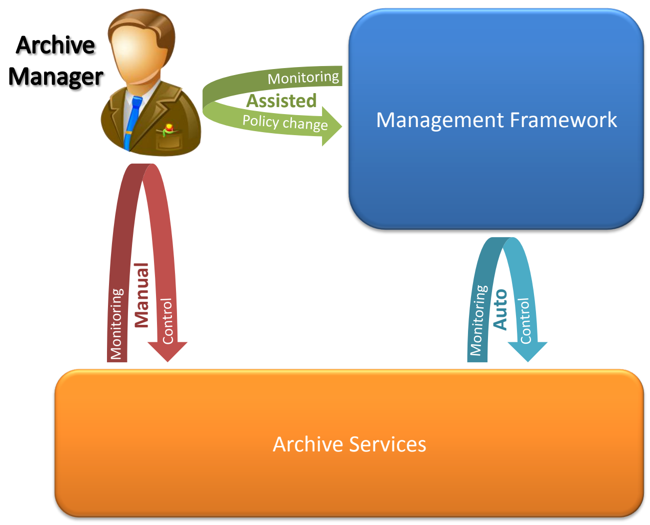

Figure 32: Management Interactions with Archive Services

derlying resources. This process represents an automated management loop (marked as 'auto' in Figure 32), in which the management framework takes action as and when required by management policy. The management policy is defined by the archive operator and may be dynamically updated. In these cases, governance components will be able to take action autonomously — e.g. to manage 'local' issues such as the allocation of CPU time to maintain ingest performance while also performing routine migration, or the selection of alternative storage services where there are redundant resources available. This automated management loop is shown on the right-hand side of Figure 32.

In addition to the above, the governance components may conclude (according to the management policy) that some action is required that cannot be implemented autonomously, such as approaching the limits of current storage capacity. In such cases a signal is sent to the human operators. These signals may simply advise the operator that management action may be needed, leaving the human to decide what action to take (if any). In some cases, the signal may also propose the action, but leave the human to decide whether or how to carry out this action. This provides an 'assisted' management loop, which is also shown in Figure 32.

The archive operator provides control inputs to mangement framework, e.g. to change the models it uses to analyse the archive's performance, or to change the range of monitoring inputs or automatic actions available to it. These control inputs do not directly affect the infrastructure itself, but do change the way agile service composition models are used to support future preservation actions. It is important to recognise that this facility to define models and policies is relevant even before the archive infrastructure and associated ICT is deployed, as well as during its operation. In the pre-deployment phase, this interaction can be used to support the ICT implementers, helping them to design and configure interconnected ICT systems in a way that allows risks affecting the preservation of data to be managed by design as well as through subsequent adaptation.

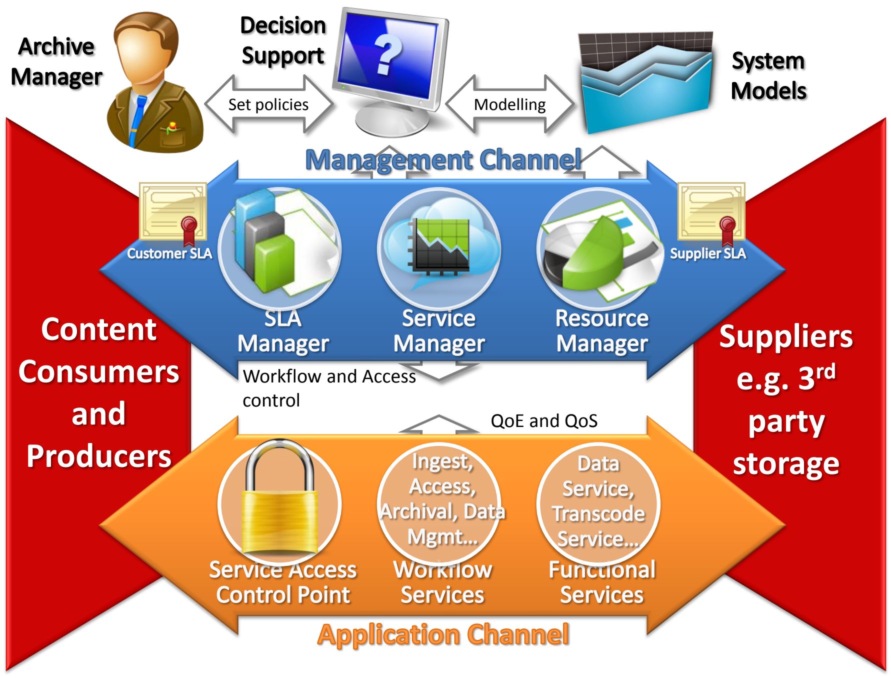

Figure 33: High-level Management Framework Architecture

5.3.3 An Architecture for SLA-based Management

Figure 33 shows a high-level view on an architecture for SLA-based management that enables the flexible system management approach described above. This architecture was developed under the SERSCIS[8] project [99] and the implementation of the component technologies is currently in progress. In SERSCIS, the management of critical infrastructure ICT components and interconnections is addressed by treating all ICT components as services, whose dependability can be specified via machine-understandable SLAs. This allows automatic and semi-automatic management of ICT dependability and interdependence. The architecture in Figure 33 comprises the following three layers:

1. The application layer at the bottom containing the functional services (ingest, access, migration, etc.).
2. The automatic management layer in the middle that takes care of the automatic management of the services and also of the customer and supplier relationships.
3. The manual management layer at the top where real people use decision support tools to set the policies of the automatic management layer. The decision support tools use models of the services which may be updated by real-world experience.

Figure 34 shows the concentric management loops that arise from the above three-layered architecture. In a manner similar to Kramer and Magee's model for autonomic

[8]European Community's Seventh Framework Programme project SERSCIS (Semantically Enhanced Resilient and Secure Critical Infrastructure Services) grant agreement n° 225336

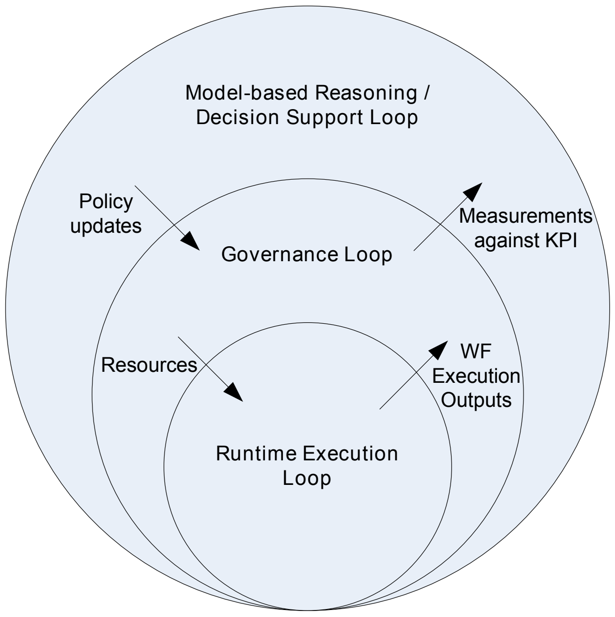

Figure 34: Concentric Management Loops

systems [100], this allows runtime information to propagate up to system decision makers and governance actions to flow down to control service execution. The inner runtime loop runs at the speed of the functional services, selecting the most appropriate resources and executing the workflow. The governance loop monitors and manages resources against SLA commitments, at a slower rate. This loop has to be asynchronous with respect to application responses because it is often infeasible for the orchestrator to wait while it verifies that the service request is in line with the customer's SLA. Finally, the outer loop runs even more slowly so that archive operators can be involved, making changes to autonomic governance policies in response to changes in key performance indicators (KPI).

Decision support and model-based reasoning tools are used to support human decision-makers. The models (and tools used to process, query and execute them) can be diverse, but are broadly broken down into three levels of system governance:

- Strategic: supporting long-term preservation planning activities; taking into account trends in content, media and formats, this should set policies on the procurement of resources, migration strategies, etc. The models used here principally support an inductive style analysis, asking "what if...?" questions to set the business goals for the archive.
- Tactical: supporting medium-term decisions that influence the archive management policy, such as which preservation actions to prioritise when demand on access or ingest reaches maximum capacity.
- Operational: supporting the day-to-day running of the archive; highlighting where faults, failures and attacks have occurred. The models and tools used here should support a deductive style of analysis, revealing the reasons for under-performance and where efforts should be focused on maintaining or upgrading the infrastructure.

The separation of decisions at each layer of Figure 34 simplifies the work that each set of components must do and is fundamental to this approach to archive management. Such a layered approach to the architecture is taken in recognition of real-world business practices and performance. It is important to note that, in this model of asynchronous service management, it is possible for a consumer of a service to breach an SLA before any action is taken to remedy this. This allows a more flexible approach to service management, which results in greater service availability and performance for the end-user, given that the user can still access the service even if the management layer is unavailable or overwhelmed. Furthermore, the consumer has (in the SLA) all the information required to regulate service usage at the client side, and should be aware of any penalties that apply for breaching this agreement.

5.3.4 Governance Components

In the architecture proposed above the management system has three components:

SLA manager deals with customer SLAs (those of the consumer and producer) which set out the constraints and service level objectives (SLOs) on ingest and access.

Resource manager deals with supplier SLAs (such as out-sourced storage or compute facilities) and with in-house resources.

Service manager balances commitments to customers with resources available internally and from external suppliers.

SLA Manager

The SLA Manager hosts a number of SLA templates that describe the parameters and levels of service that the preservation service provider is willing to negotiate. These are encoded as a set of metrics and constraints on these metrics. ID3.4.1/Section 3 describes a number of such metrics. The SLA manager handles requests from clients (content producers and consumers) for SLAs based on the published templates.

On successfully negotiating a new SLA, the SLA manager can be contacted to provide information to the clients on the status of their SLA (QoS measurement reports). Notifications of activities relating to SLA are communicated to the service manager who may require that existing SLAs be terminated or suspended if required to maintain overall system QoS.

Resource Manager

The Resource Manager handles the acquisition, allocation and removal of resources, both hosted 'in-house' by the archive and provided externally by third-party suppliers.

The resource manager maintains a registry of these resources, comprising the address for accessing the service, the SLA describing the service 'capacity', and accumulated reports of Quality of Experience (QoE) from using the service.

Service Manager

The service manager must balance the commitments and resources to meet the terms of the SLAs that are in force whilst always maintaining the safety of the data. This component is an event-decision-action loop, where the decision is made according to a policy. The 'intelligence' can be provided by a combination of automatic and manual analysis supported by system modelling tools. The output of this analysis is the management policy, by which the service manager decides which action to take to meet its commitments, plan to continue meeting them, and mitigate the effects of events (e.g. failures) that cause it to stop meeting them.

5.3.5 Services as Workflows

Many services will be provided as a workflow of other services. The preservation workflow orchestration component can use the registry maintained by the resource manager to discover resources when it has to execute a workflow. If the services are functionally identical, the orchestrator can use this fact to provide a workflow service that meets the QoS commitments to its customers in the following ways:

- selecting a workflow that encompasses the steps required to meet the customer's SLA;
- selecting services whose individual QoS combine to meet the end-to-end QoS commitments of the workflow;
- reselecting among alternative services in the event of faults, failures and under-performance.

These workflow strategies are explained in more detail below.

Selecting the Right Workflow

An archive may offer several SLA templates that describe varying levels of QoS to its customers. Typically, increasing levels of QoS entail more elaborate workflows. For example, a 'bronze' SLA may offer to store content and provide a bit-level identical copy on demand. The workflows for preservation may only require periodic fixity checks in this case. A 'silver' SLA may (in addition) offer to ensure that the file structure is correct, requiring a step in the ingest workflow to check that the MXF container structure is correct. Finally, a 'gold' SLA may offer to ensure content usability, entailing additional steps to provide AV quality assessment on ingest, e.g. to detect problems such as frame drop-outs.

Selecting the Right Services (Dynamic Composition of Services)

Once the appropriate workflow has been chosen, the selection of the services used in each step of the workflow determines the end-to-end QoS of the workflow. The workflow orchestrator can dynamically compose a chain of services whose individual characteristics combine to meet an agreed SLA. For example, if a high-priority customer requires that multiple users from their organization have concurrent random access to content with low latency (as agreed in their SLA), the orchestrator may choose to store the data on disk rather than on data tape. Similarly, if a content producer requires a high degree of availability, the ingest workflow may replicate the content across in-house and external storage services.

Dynamically Reselecting Services

The workflow orchestrator can also provide a high degree of robustness by dynamically adapting to faults and failures in the workflow through reselecting alternative services. For example, if a storage service fails the integrity check, the orchestrator can select an alternative source for the content (if one exists). Services other than storage may be more readily interchangeable, given that their failure does not necessarily render the content inaccessible.

Preservation Workflows

D2.2.1 contains an extensive range of workflows involved in preservation, some of which may be automated. The archive is focused on three main data-centric processes, each of which may be executed as a workflow:

1. Ingest
2. Access
3. Preservation actions

The preservation actions could range from periodic fixity checks (disc scrubbing) to large scale planned format or hardware migrations. In terms of priority for access to resources one could argue that preservation actions are most important, then ingest (as the data to be ingested may not be considered safe until in the archive) and then access. The balance is not that simple however: a large format migration might take weeks even if all resources were used but it would be impractical to shut down all access to the archive during that period and though a migration might be required for long-term safety, doing it a little more slowly would not increase the immediate threats.

In order to manage the constraints and service level objectives (SLOs) of the agreed SLAs, the automatic management system needs to both monitor and manage the services underneath. Some information on monitoring and management mechanisms is set out in ID3.4.1/Section 3. A robust architecture would decouple the management and the service as much as possible, for instance by the services being autonomous. This can be achieved by the services understanding the parts of the policies (such as security policies) relevant to them and therefore able to make their own immediate decisions. The monitoring and management loop can then be asynchronous with the service manager requesting usage reports (either queued or instantly generated) from the services periodically (as a pull) and then updating the services' policies as necessary in a slower time-frame. The manual management layer acts more slowly, gathering the monitoring information as necessary to update models, considering predicted trends or deciding upon a proposed migration and then updating the policies of the service manager or procuring additional resources to implement the resulting decisions.

5.3.6 Service Management Use Cases

Components of the System

A use case analysis documents the interactions of various actors with a "system". In this case we will consider the "system" to be all three management components: the SLA service, the resource manager, and the service manager. The interactions between these three components are therefore not documented here. The use cases are grouped in the diagram and in this text according to the component they interact with.

The architecture manages local functional services, each deployed in the context of an SLA. Some services, called "workflow services" make use of other services, either local or remote, during their execution. Workflow services are distinguished here because they report the quality of experience (QoE) that they receive from the services they use. A workflow service does not have to be a service that executes BPEL or similar workflow language, it is any local service that uses another service of some kind.

Actors

The actors interacting with the system defined above are:

- Service administrator: oversees the running of the management system.
- Resource administrator (local): deals with the provision of the necessary local and remote resources to provide the archive service.
- Customer administrator deals with producers and consumers (the users of the archive). The word "customer" does not necessarily imply payment for use of the service.
- Decision support tool suite: monitors the service management system and provides different views to support the work of the service administrator, resource administrator (local) and customer administrator.

- Local service (e.g. a storage service): reports QoS metrics and is managed by the system.
- Workflow service (is a local service): a more complex local service that also uses other services (both local and remote).
- Service user: client who uses the archive (producer or consumer). This “client” could be a person or a piece of software (e.g. a remote service).
- Resource administrator (remote) (is a service user): negotiates and controls SLAs at the client organization.

There are two “resource administrators” shown on the diagram and listed here. They are the same actor but in two different organizations and so interact with the system in different ways.

Use Case Diagram

The following use case diagram 35 unconventionally groups sets of related use cases to aid readability and also to show the component that the use case relates to.

Use Cases

The following use cases are grouped by the service management component and are just outlines, briefly describing the use case and the principal actor(s).

Resource Manager The resource manager keeps track of supplier SLAs (such as outsourced storage or compute facilities) and with in-house resources. It gathers QoE reports on the use of services.

- **Add/Remove Local Resources**

  Resource administrator (local) defines the capacity provided by the locally hosted services (e.g. quantity of disc space).

- **Add/Remove Supplier SLA**

  The resource administrator (local) adds and removes SLAs with third parties that provide additional capacity. Prior to doing this, the resource administrator (local) will have used a decision support tool to determine what remote capacity is (or will be) required and proposed the necessary SLA(s). These actions do not interact directly with the system being modelled here though so are only mentioned in this text.

- **Query Capacity**

  Decision support tool queries what the current capacity is, this includes the capacity provided by both local and remote services.

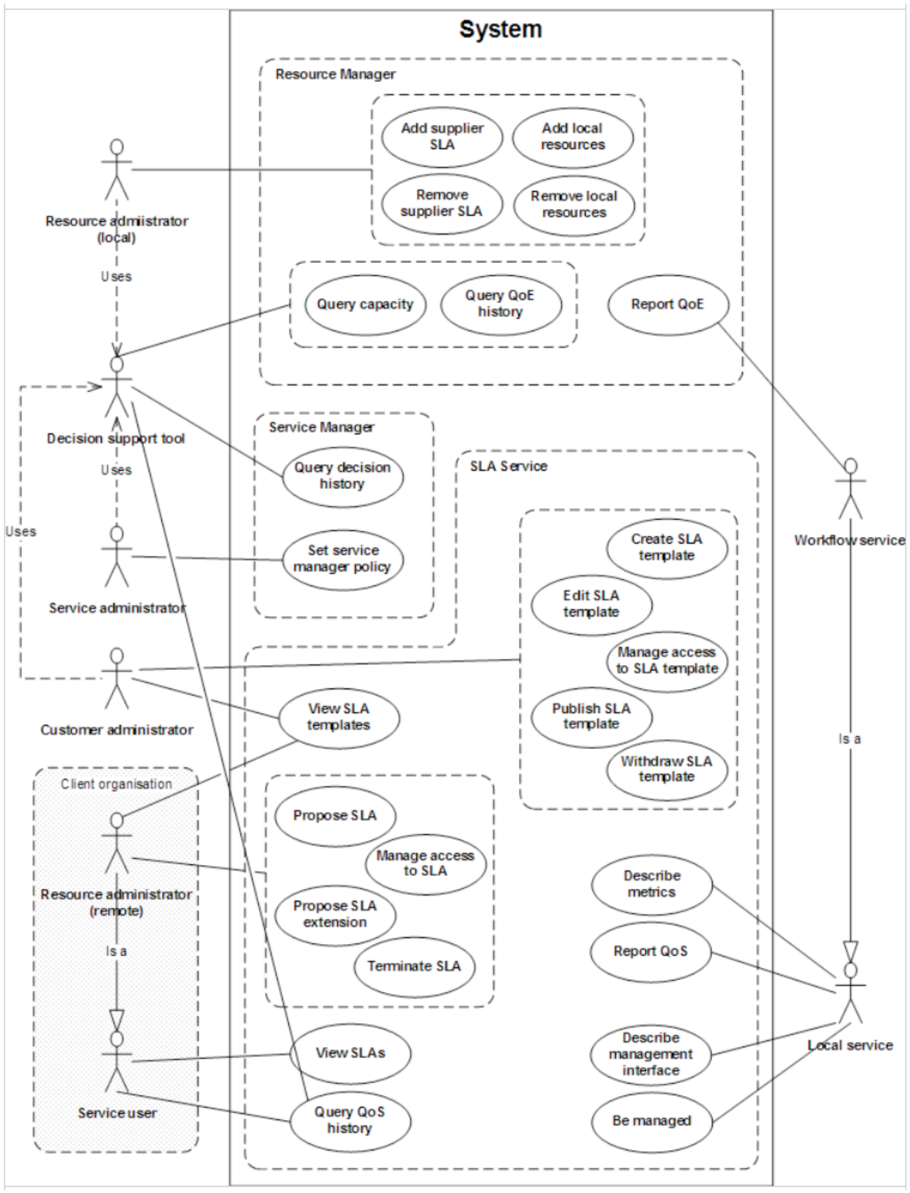

Figure 35: Use case diagram for the service management system. The named dashed blocks inside the “System” divide the use cases amongst the three components of the service management system. Some use cases are further divided into logical groups to reduce the number of lines from actors to use cases.

- **Report QoE**

  A local workflow service reports the QoE it has had from using both local and remote services.

- **Query QoE History**

  Decision support tool queries the QoE records of both local and remote services.

Service Manager The service manager monitors both the commitments made to customers (producers and consumers) and the resources available internally and externally. It is configured by a policy to automatically take actions to maintain a balance of resources and commitments where possible.

- **Query Decision History**

  Decision support tool queries to get list of the decisions the service manager has made (what instruction was issued and why).

- **Set Service Manager Policy**

  The assisted operator can view and change the service manager's policy.

SLA Service The SLA service provides the interface for the customers (producers and consumers) to manage their SLA(s).

- **Create SLA Template**

  The customer administrator creates a new SLA template (offer).

- **Edit SLA Template**

  The customer administrator, using information provided by a decision support tool, edits the SLA template, setting the SLA terms.

- **Publish SLA Template**

  The customer administrator publishes the template, making it live. The service manager will then include the SLA template in its decisions.

- **Manage Access to SLA Template**

  The customer administrator controls who can use the SLA template.

- **Withdraw SLA Template**

  The customer administrator withdraws the SLA template, meaning it is no longer available and is not taken into account in any policy decisions.

- **View SLA Templates**

  Budget holders and the customer administrator browse the published SLA templates to see what offers are available. The resource administrator (remote) may not see all the SLA templates as they must first be given access.

- **Propose SLA**

  The resource administrator (remote) chooses an SLA template and proposes it to create a new SLA. The system decides if it agrees with the proposal and accepts or rejects the proposal accordingly.

- **Manage Access to SLA**

  The resource administrator (remote) controls who can use the SLA.

- **Propose SLA Extension**

  The resource administrator (remote) requests a change to an existing SLA. The extension might be to change the resources or the time period.

- **Terminate SLA**

  The resource administrator (remote) terminates an existing SLA. This has the effect of terminating access to any services deployed under the SLA.

- **View SLAs**

  A service user requests a list of SLAs. The ones that the user is permitted to see are displayed.

- **Query QoS History**

  A service user or the decision support tool asks for usage information. The query may be in the context of an SLA or SLAs, involve a time period or a certain set of metrics. The user can only query those SLAs that he/she has access to.

- **Describe Metrics**

  A local service describes the metrics it will later report usage on using an ontology.

- **Describe Management Interface**

  A local service describes the interface that can be used to manage it. For instance, the interface might be a mechanism for setting the value of certain keys to change the service state or control other parameters. The keys and possible values would be described so that they could be used in SLA and policy definitions.

- **Report QoS**

  A local service reports the QoS it has provided to a service user or another service. These reports are in the context of an SLA using the metrics it has previously described, e.g. disc space usage, uptime, bit errors.

- **Be Managed**

  A local service can be managed using the interface it describes.

Local Services The consequence of this design on local services that are to be managed is two-fold:

1. A service must provide a usage reporting interface which can also describe the terms it uses via an ontology.
2. A service must provide a management interface which also can describe the management options available.

5.4 License and support

As mentioned, the PrestoPRIME Preservation Platform will be released according to an open license. More precisely it will be the software implemented in D5.2.2 [3] where the Platform Specific Model (PSM) of the reference architecture (PIM) is provided. The related documentation and source code will be available publicly at the project website. A great advancement will be the planned *Competence Center*, that will provide support to all the users that need information and assistance in setting up digital repositories and preservation systems or that want to experiment the open platform [3] as well as the commercial system [5, 6].

Even though the core framework will be freely available, the many software components created in the different Work Packages will have their own license: not a single open source license will cover the overall platform but more likely a set of compatible licenses, OSI [101] recognized and assessed. Moreover some software components can be *propriety* of the partner developing it, shared to the other partners during the life time of project according to our Consortium Agreement and released under a commercial license.

Before the end of the project and when with the development of [3] and [5, 6] the specific license of all the involved components will be provided and published.

6 Conclusions

This deliverable presents the overall architecture of the PrestoPRIME Preservation Platform. A Section on the State of the Art takes into account similar international projects dealing with the preservation systems. Then, there are two Sections on Submission Information Package: one for a general discussion and analysis of examples from other projects and institutions, and one for a candidate structure for the SIP. Finally the architecture is described from a software point of view.

Section 5.2 describes the PrestoPRIME Preservation Platform **design**, pointing out the model driven approach, where the need of a specific behavior introduces a preexisting design pattern solving it. For each functional entity listed in the OAIS Figure 22 a software component diagram has been analysed and designed. Summing up all the used patterns and the components solving specific problems (and realising specific functionalities), the architecture comes up consequently with an overall view reported in Figure 23.

Starting from the OAIS model and taking into account the best practices designing one of the leaders commercial preservation systems (Rosetta [4]), this document generalises the required software components in order to implement a complete preservation system, providing the reference architecture of the PrestoPRIME Preservation Platform.

In order to perform the preservation tasks (core functionality of a preservation system) section 5.3 proposes a SLA based management approach, taken into account in the architecture at the *Administration* level, but that will impact the overall software components described.

The following Table 10 shows a comparison of the *Scenarios* [1] with the *Reference Architecture* 5.2. First column lists the scenarios identified in [1]. Second column named *Relevance for architecture* reports how much the proposed architecture meets the related needs. The table aims at show that the proposed architecture and SIP structure does address project requirements. For the abstract level of the current specification, most of the software will be defined in the specific implementations; nevertheless it is important to verify that all the required components (even if *abstract*) have been covered.

Scenario	Relevance for Architecture
Preservation for Broadcast Archives	The required AV formats are covered by SIP Section 4. Storage is described in Section 5.2.4 and the *StorageHandler* component is able to deal with different implementation. A candidate specific software implementation could also make use of external libraries such as Fedora [18]. The *decision support* is undertaken by the PreservationPlanning package 5.2.3 where components such as the PreservationPlanner and Evaluator combined with the RiskAnalyzer can suggest to the user the action to perform. Also the *ServiceManagement* component in the Adminisration 5.2.6 package is responsible for suggesting actions, deeply detailed in SLA Section 5.3. Digitization and Ingest are covered in SIP Section 4, SLA Section 5.3 and by the software components in the Ingest Section 5.2.1 where the *IngestFrontController* is responsible to handle and dispatch ingest requests to the specific components.
Rights Clearance for Access and Fruition	This scenarios is mainly covered by Work Package 4 Task 4. The described SIP 4 is able to handle complex contracts mapping as well as complex rights definitions. The Access 5.2.5 package has the *DIPFactory* which is able to disseminate the same rights ingested. The preservation of rights metadata is a pure implementation issue. In the Access 5.2.5 package the *AccessFrontController* will dispatch the requests to the right software component and must give back the responses according to the rights associated to the searched item. We can plan to have a specific component for this purposes in the specific implementation. Rights updates are supported with a new SIP submission and in the specific implementation a version system should be implemented as partially reported in the DataPersistenceLayer Section **??**, where an AIPHistory component is shown.
The perspective of other organizations with AV archives	With respect to the Higher Education Institution (HEI), the Architecture is quite well aligned. HEIs require to support more content formats (coming from a Library context) then those identified in the broadcasting environment, but it will be a matter of specific implementations of the *SIPMetadataEnrichment* 25 component and also the implementations of the Storage component under the *StorageHandler* 28. It could be needed to handle different SIP structures. It can be met within the specific implementation of the *Ingest* package.

Scenario	Relevance for Architecture
	Concerning the Small and Medium Archives, the requirements are more related to the business model and can be addressed with specific implementation of the *ServiceManager* reffig:administration and the *SLA-based Management* 5.3 software components. Other requirements of Small Medium Archives will be covered within Work Packages six and seven.
Adding AV-content to Europeana and improving its usability	We can imagine several approaches for transmitting information to Europeana: • the simplest can be considered as an external module getting the DIP provided by the *AccessFrontController* 29. The external module should be able to translate the DIP into the format required by Europeana. • we can place this module into the Europeana services, pulling DIP directly from the *AccessFrontController*. • the *AccessFrontController* can implement the needed format required by Europeana answering to the requests accordingly. This is a matter of specific implementation of the software component. Anyway the *AccessFrontController* will implement several protocols. In Figure 29 it is written “WS” that means *WebServices* generically, but at least the OAI-PMH protocol should be supported in the specific implementation. As stated in Section 5.1 the specific implementation can benefit from the adoption of some tool already available such as the *OAI-PMH Data Provider* [17]. What is missing and will be better analyzed and described within Work Package 4 Task 4 is the issue related to the Rights associated to a specific content and in this scenario, how to map complex BusinessToBusiness Rights (coming from broadcasting or library environments) into a much more simplified BusinessToConsumer schema (which is the *internet* “access and fruition” model).

Scenario	**Relevance for Architecture**
Information enrichment for Access and Fruition	In this scenario a big issue is the *validation* of the metadata provided by the users. The *SIPApprover* in Figure 25 component can be implemented (inheriting) as *Manual* or *Automatic* according to the need of the supervisor interaction or not for validation. Digital Preservation requires to maximize the “automation” and the specific implementation will make use of the available tools minimizing the human being actions. A requirement arising from this scenario is the frequent “update” of the AIPs. Once again it is a matter of specific implementation of the *Ingest*, *Data Management* and *Storage* packages; in Section **??** a candidate abstract internal representation of AIPs is provided but a specific implementation can adopt different solutions. Anyway a further analysis of the descriptive metadata structure for the SIP and its update will be defined in Work Package 2 [52] and for what concerns the Rights, will be formalized in Work Package 4 Task 4.
The perspective of Service Providers	Since the proposed architecture is based upon the OAIS logic with a software design approach, it has already towards a Service Oriented approach. The requirements arisen in this scenario are more precisely fulfilled by the SLA Section 5.3 and its software components *SLAManager*, *ServiceManager* and *ResourceManager*. The specific software implementation will meet the requirements.

Table 10: Comparison of *Scenarios* [1] with the *Reference Architecture* 5.2

From the exposed reference architecture, two compliant implementations will be developed in the next months:

- the open source PrestoPRIME preservation platform (named P4), that will be described in D5.2.2 [3]
- the commercial PrestoPRIME version of Rosetta, that will be described in D5.3.1 and D5.3.2 [5, 6]

As previously declared in this document, the architecture as well the IP (Information Package) organization is not to be considered frozen and exhaustive for the preservation platform, neither for the software implementation. It should be taken as a guideline and the PrestoPRIME attempt to translate the OAIS specification into software components. Due to the complexity of the system considered in this deliverable (a complete preservation system), some minor parts have not been covered. If, during the implementation phase, some components will be structurally changed or if something will be added in order to have a better design, an amendment of this document will be delivered.

References

[1] PrestoPRIME Project. Deliverable 5.1.1: Definition of Scenarios.

[2] OMG. Unified Modeling Language uml 1.4.2 available as iso/iec 19501; 19505 assigned to uml 2.1.2; itu-t recommendations z.100 (sdl) and z.109 (sdl uml profile). http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML. Last visited: 2010 May 01.

[3] PrestoPRIME Project. Deliverable 5.2.2: Prototype of open prestoprime reference implementation.

[4] ExLibris. Ex Libris Rosetta: Preserve your digital assets. http://www.exlibrisgroup.com/files/Products/Preservation/Rosetta-A4LOWres.pdf. Last visited: 2010 May 01.

[5] PrestoPRIME Project. Deliverable 5.3.1: Proof of concept of the capability of exlibris commercial system to provide intarfacing mechanism compliant to the prestoprime reference architecture.

[6] PrestoPRIME Project. Deliverable 5.3.2: Complete implentation of exlibris commercial system integrating the outputs of wp3 and wp4 in in compliance to the prestoprime reference architecture.

[7] ExLibris. ExLibris: The bridge to Knowledge. http://www.exlibrisgroup.com/. Last visited: 2010 May 01.

[8] CCSDS. Reference Model for an Open Archival Information System (OAIS), Blue Book. http://public.ccsds.org/publications/archive/650x0b1.pdf. Last visited: 2010 May 01.

[9] Center for Research Libraries CRL. Trustwothy repository audit and certification. http://catalog.crl.edu/record=b2212602~S1. Last visited: 2010 June 14.

[10] CASPAR Project. Homepage. http://www.casparpreserves.eu/. Last visited: 2010 May 01.

[11] Planets Project. Preservation and Long-term Access through NETworked Services. http://www.planets-project.eu. Last visited: 2010 May 01.

[12] SHAMAN Project. SHAMAN: Sustaining Heritage Access through Multivalent ArchiviNg. http://shaman-ip.eu/shaman/. Last visited: 2010 May 01.

[13] KEEP Project. Keeping Emulation Environments Portable. http://www.keep-project.eu/ezpub2/index.php. Last visited: 2010 May 01.

[14] Preserv 2. Homepage. http://preserv.eprints.org/. Last visited: 2010 May 01.

[15] KeepIt. Homepage. http://preservation.eprints.org/keepit/. Last visited: 2010 May 01.

[16] JISC. Homepage. http://www.jisc.ac.uk/. Last visited: 2010 May 01.

[17] DSpace project. Homepage. http://www.dspace.org/. Last visited: 2010 May 01.

[18] Fedora Commons. Homepage. http://www.fedora-commons.org/. Last visited: 2010 May 01.

[19] CCSDS. Reference Model for an Open Archival Information System (OAIS), Pink Book. http://public.ccsds.org/sites/cwe/rids/Lists/CCSDS%206500P11/Attachments/650x0p11.pdf. Last visited: 2010 May 01.

[20] CASPAR Project. D2101: Prototype OAIS-Infrastructure. http://www.casparpreserves.eu/publications/deliverables. Last visited: 2010 May 01.

[21] CASPAR Project. CASPAR CONCEPTUAL MODEL - PHASE 1. http://www.casparpreserves.eu/Members/cclrc/Deliverables/caspar-conceptual-model-phase-1-1/at_download/file. Last visited: 2010 Mar 22.

[22] CASPAR Project. CASPAR OVERALL COMPONENT ARCHITECTURE AND COMPONENT MODEL. http://www.casparpreserves.eu/Members/cclrc/Deliverables/caspar-overall-component-architecture-and-component-model-1/at_download/file. Last visited: 2010 May 01.

[23] CASPAR Project. CASPAR - Overall Architecture, Components and Interfaces. http://www.casparpreserves.eu/Members/cclrc/Deliverables/caspar-overall-architecture-components-and-interfaces/at_download/file. Last visited: 2010 May 01.

[24] XAM. eXtensible Access Method. http://www.snia.org/tech_activities/standards/curr_standards/xam/. Last visited: 2010 May 01.

[25] OSD. Object store device. http://www.haifa.il.ibm.com/projects/storage/objectstore/index.html. Last visited: 2010 Jan 26.

[26] Ross King. ERCIM News - The Planets Interoperability Framework. http://ercim-news.ercim.eu/en80/special/the-planets-interoperability-framework. Last visited: 2010 Mar 29.

[27] Planets Project. Planets Testbed. http://testbed.planets-project.eu/testbed/. Last visited: 2010 May 01.

[28] Planets Project. PLATO - The Preservation Planning Tool. http://www.ifs.tuwien.ac.at/dp/plato/intro.html. Last visited: 2010 May 01.

[29] Planets Project. Planets IF2345 - D3 Release Report Final. http://www.planets-project.eu/software/Planets_IF2345-D3_ReleaseReport-Final_Website.pdf. Last visited: 2010 May 01.

[30] Acegi security system. http://www.acegisecurity.org/. Last visited: 2010 June 7.

[31] SHAMAN Project. D2.3 Specification of Reference Architecture. http://shaman-ip.eu/shaman/sites/default/files/SHAMAN_D2.3-Specification%20Reference%20Architecture.pdf. Last visited: 2010 May 01.

[32] iRods. Integrated Rule-Oriented Data System. http://irods.org/pubs/iRODS_Fact_Sheet-0907c.pdf. Last visited: 2010 May 01.

[33] SRB. DICE Storage Resource Broker. http://www.sdsc.edu/srb/index.php/Main_Page. Last visited: 2010 May 01.

[34] PREMIS. PREMIS Implementation Registry. http://www.loc.gov/standards/premis/premis-registry.php. Last visited: 2010 May 01.

[35] KEEP Project. KEEP Relationship with other EU projects. http://www.keep-project.eu/ezpub2/index.php?/eng/About-KEEP/Technical-solution/Relationship-with-other-EU-projects. Last visited: 2010 May 01.

[36] EPrints. Homepage. http://www.eprints.org/. Last visited: 2010 May 01.

[37] Open Archives Initiative. Open archives initiative object reuse and exchange. http://www.openarchives.org/ore/. Last visited: 2010 June 07.

[38] National Archives. The technical registry PRONOM. http://www.nationalarchives.gov.uk/PRONOM/Default.aspx. Last visited: 2010 May 01.

[39] EPrints. ROAR Registry of Open Access Repositories. http://roar.eprints.org/. Last visited: 2010 May 01.

[40] National Archives. DROID Digital Record Object Identification. http://droid.sourceforge.net/. Last visited: 2010 May 01.

[41] PREMIS. Preservation Metadata Maintenance Activity. http://www.loc.gov/standards/premis/. Last visited: 2010 May 01.

[42] DSpace project. DSpace Manual. http://www.dspace.org/1_6_0Documentation/. Last visited: 2010 May 01.

[43] Corporation for national research Intitiatives. The Handle System. http://www.handle.net/. Last visited: 2010 May 01.

[44] Apache. Lucene. http://lucene.apache.org/java/docs/index.html. Last visited: 2010 May 01.

[45] Fedora Commons. Getting Started. http://www.fedora-commons.org/confluence/display/FCR30/Getting+Started+with+Fedora. Last visited: 2010 Mar 25.

[46] Fedora Commons. Service Framework. http://www.fedora.info/download/2.2.1/userdocs/server/features/serviceframework.htm. Last visited: 2010 May 01.

[47] Fedora Commons. Fedora SIP Creator. http://www.fedora-commons.org/download/2.2/services/sipcreator/doc/index.html. Last visited: 2010 May 01.

[48] Fedora Commons. Fedora Release Notes 3.0b1. http://www.fedora-commons.org/documentation/3.0b1/userdocs/distribution/release-notes.html. Last visited: 2010 May 01.

[49] Yaniv Levi. Digitlna Kninica — Jasn, Slovakia — March 2009. www.schk.sk/fileadmin/docs/digilib/2009/Yaniv-LEVI.pdf. Last visited: 2010 May 01.

[50] PrestoPRIME Project. Deliverable 3.1.1: Specification and design of a preservation environment for audiovisual content.

[51] PrestoSPACE Project. Homepage. http://prestospace.org/. Last visited: 2010 May 01.

[52] PrestoPRIME Project. Deliverable 2.2.2: Metadata Models, Interoperability Gaps and Extensions to Preservation Metadata Standards. Publicly available after June 2010.

[53] UIBK. University of Innsbruck. http://www.uibk.ac.at/. Last visited: 2010 May 01.

[54] University of Innsbruck. http://www.loc.gov/standards/mix/. Last visited: 2010 May 01.

[55] University of Innsbruck. http://www.niso.org/kst/reports/standards?step=2&gid=None&project_key=b897b0cf3e2ee526252d9f830207b3cc9f3b6c2c. Last visited: 2010 May 01.

[56] University of Innsbruck. http://www.literature.at/default.alo;jsessionid=1332DF70E895B2B3755801DF8640B70E. Last visited: 2010 May 01.

[57] University of Innsbruck. `http://meta-e.aib.uni-linz.ac.at/alo/alo.html`. Last visited: 2010 May 01.

[58] Holger Brocks et al. The shaman context model. `http://www.ogf.org/OGF28/materials/1954/Brocks_OGF28-DR-RG-18032010.pdf`. Last visited: 2010 June 07.

[59] Harvard University. JHOVE jstor/harvard object validation environment. `http://hul.harvard.edu/jhove/`. Last visited: 2010 May 01.

[60] R. D. Jones R. Sanderson. Foresite-toolkit. `http://code.google.com/p/foresite-toolkit/`. Last visited: 2010 June 07.

[61] Library of Congress. Preserving digital public television. `http://www.thirteen.org/ptvdigitalarchive/`. Last visited: 2010 june 04.

[62] Kara Van Malssen. Pbcore, mets, premis, mods, metsright...oh my! `http://www.slideshare.net/kvanmalssen/pbcore-mets-premis-mods-metsrightsoh-my`, November 2008. Association of Moving Image Archivists Conference, Savannah, GA.

[63] Joseph Pawletko. Repository design report with attached metadata plan. Technical report, National Digital Information Infrastructure and Preservation Program Preserving Digital Public Television Project, 2010.

[64] Nan Rubin. Preserving digital public television: Is there life after broadcasting? In *International Preservation News - A Newsletter of the IFLA Core Activity on Preservation and Conservation*.

[65] MXF. Material Exchange Format. `http://rfc-ref.org/RFC-TEXTS/4539/kw-mxf.html`. Last visited: 2010 May 01.

[66] Microsoft. AVI: Audio Video Interleave. `http://tools.ietf.org/html/rfc2361`. Last visited: 2010 May 01.

[67] Apple. MOV: Object Movie File Format. `http://developer.apple.com/legacy/mac/library/technotes/tn/tn1036.html`. Last visited: 2010 May 01.

[68] Society of Motion Picture and Television Engineers. Smpte 377 - television . material exchange format (mxf) file format specification (standard). *SMPTE Standards*, 2004.

[69] MXFLib - A C++ Library for MXF file I/O. `http://sourceforge.net/projects/mxflib/`. Last visited: 2010 May 17.

[70] PrestoPRIME Project. Deliverable 2.1.3: AV Data Model: Final Specifiaction. Publicly available after December 2010.

[71] METS. Metadata Encoding and Transmission Standard. http://www.loc.gov/standards/mets/. Last visited: 2010 May 01.

[72] METS. METS Primer and Reference Manual. http://www.loc.gov/standards/mets/METSPrimerRevised.pdf. Last visited: 2010 May 21.

[73] MDA. Model Driven Architecture. http://www.omg.org/mda/. Last visited: 2010 May 01.

[74] OMG. Object Management Group. http://www.omg.org/. Last visited: 2010 May 01.

[75] David Chappell. *Enterprise Service Bus*. O'Reilly Media, Inc., 2004.

[76] OAIS Organization for the Advancement of Structured Information Standards. Service Oriented Architecture. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm. Last visited: 2010 May 01.

[77] SOAP. Simple Object Access Protocol. http://www.w3.org/TR/soap/. Last visited: 2010 May 01.

[78] WSDL. Web Services Description Language. http://www.w3.org/TR/wsdl. Last visited: 2010 May 01.

[79] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Dissertation for the doctor of philosophy in information and computer science, University of Califonia, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf – geprüft: 19. März 2004.

[80] UDDI. Universal Description, Discovery, and Integration. http://www.oasis-open.org/specs/index.php#uddiv3.0.2. Last visited: 2010 May 01.

[81] BPEL. Business Process Execution Language. http://www.ibm.com/developerworks/library/specification/ws-bpel/. Last visited: 2010 May 01.

[82] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: Abstraction and reuse of object-oriented design, 1993.

[83] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[84] Erich Gamma. Design patterns - 15 years later. In *ECOOP*, page 1, 2006.

[85] Kent Beck and Ralph E. Johnson. Patterns generate architectures. In *ECOOP*, pages 139–149, 1994.

[86] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, 1977.

[87] Christopher Alexander. *The Timeless Way of Building*, volume 1 of *Center for Environmental Structure Series*. Oxford University Press, New York, NY, 1979.

[88] Deepak Alur, John Crupi, and Dan Malks. *Core J2EE Patterns: Best Practices and Design Strategies, 2nd Edition*. Prentice Hall, Sun Microsystems Press, 2003.

[89] UUID. Universally Unique Identifier. `http://java.sun.com/j2se/1.5.0/docs/api/java/util/UUID.html`. Last visited: 2010 May 01.

[90] UMID. Unique Material Identifier. `http://www.iptvdictionary.com/IPTV_Dictionary_DRM_Unique_Material_Identifier_UMID_Definition.html`. Last visited: 2010 May 01.

[91] New Zealand National Library. Metadata extraction tool. `http://meta-extractor.sourceforge.net/`. Last visited: 2010 May 01.

[92] EJB3. Enterprise JavaBeans Technology. `http://java.sun.com/products/ejb/`. Last visited: 2010 May 01.

[93] JPA. The Java Persistence API. `http://java.sun.com/developer/technicalArticles/J2EE/jpa/`. Last visited: 2010 May 01.

[94] DAO. Data access object. `http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html`. Last visited: 2010May 01.

[95] Kent Beck, Erich Gamma, and David Saff. Junit documentation, cookstour. `http://junit.sourceforge.net/doc/cookstour/cookstour.htm`. Last visited: 2010 May 01.

[96] OAI-PMH. OAI-PMH Protocol for Metadata Harvesting. `http://www.openarchives.org/pmh/`. Last visited: 2010 May 01.

[97] MPQF. MPEG Query Format (ISO/IEC 15938-12). `http://www.fim.uni-passau.de/en/fim/faculty/chairs/chair-of-distributed-information-systems/research/mpeg-query-format-mpqf.html`. Last visited: 2010 May 01.

[98] JPSearch3. Overview of JPSearch: a Standard for Image Search and Retrieval. `http://infoscience.epfl.ch/getfile.py?docid=14159&name=2007_cbmi_dufaux_et_al&format=pdf&version=1`. Last visited: 2010 May 01.

[99] M. Hall-May and M. Surridge. Resilient critical infrastructure management using service oriented architecture. *International Workshop On Coordination in Complex Software Intensive Systems (COCOSS)*, 2010.

[100] J. Kramer and J. Magee. A rigorous architectural approach to adaptive software engineering. *ournal of Computer Science and Technology*, 2009.

[101] OSI. Open source initiative. `http://www.opensource.org/`. Last visited: 2010 Jun 29.