FP7-ICT-2007-3-231161

Internal Deliverable ID3.1.1

Specification and Design of a Preservation Environment for Audiovisual Content

Nir Kashi (ExLibris)

18/05/2010

Document administrative table

Document Identifier	PP_WP3_ID3.1.1_preservation_specification_R1 Release 1
Filename	PP_WP3_ID3.1.1_preservation_specification_R1_v2.01.doc
Workpackage and Task(s)	WP3 Data management and processing for media preservation WP3T1 Processing and workflows for audiovisual migration
Authors (company)	Nir Kashi (ExLibris)
Contributors (company)	Tamara Friedman (ExLibris)
Internal Reviewers (company)	Christoph Bauer (ORF), Marcel Mattheijer (B&G)
Date	2010-05-18
Status	Release
Type	Part of Deliverable
Deliverable Nature	R - Report
Dissemination Level	PU - Public
Planned Deliv. Date	31 December 2009
Actual Deliv. Date	24 February 2010
This IsPartOf	Part of D3.1 Design and specification of the audiovisual preservation toolkit
This HasPart	-
Abstract	This report presents the specifications and designs for a framework for a long term preservation system for audiovisual content using migration, based on the OAIS reference model, taking into account the research done for audiovisual needs and also ExLibris' experience in this area. Highlights of the research include well known standards such as OAIS, PREMIS, & METS as well as a review of tools for metadata extraction and file identification. A major emphasis was placed on designing the system with the ability to integrate the tools that will be developed within the scope of the PrestoPrime project.

DOCUMENT HISTORY

Version	Date	Reason of change	Status	Distribution
v0.1	Nov 6, 2009	First Version	Outline	Confidential
v0.2	Dec 12, 2009	Technical Writer Review/Edits	Working Draft	Confidential
v1.0	Dec 16, 2010	Accept TW revisions (mainly format/style changes)	Working Draft	Confidential
v1.1	Feb 8, 2010	Additions/modifications based on comments received from reviewers	Advanced Draft	Confidential
v1.2	Feb 15, 2010	Add abstract paragraph	Final Draft	Confidential
v1.3	Feb 17, 2010	Additions/modifications based on comments received from reviewers	Final Draft	Confidential
v2.0	Feb 24, 2010	Finalised - Delivered	Release	Restricted
v2.01	May 18, 2010	Final adjustments - Published	Release	Public

Table of contents

Scope

As part of the PrestoPRIME project, Ex Libris will investigate and gain extensive knowledge in the audiovisual area and potentially enhance the Ex Libris Digital Preservation System named Rosetta. The scope will include system design, tools assessment, plug-in architecture, integration with other systems, and other items as needed.

Executive Summary

Design and specification of a long-term preservation system for audiovisual content has many aspects starting with the ingest/deposit aspects; continuing with the repository, access, and management aspects; and, at last, the preservation management that allows different approaches such as migration, emulation or multivalent
Since there are many aspects (described above) and to simplify, as much as possible, the design of a preservation system, all aspects were categorized into one of the two main sub-systems that are in a preservation system, the digital repository component and the digital preservation component.
Under the digital repository component, the work done includes:

- Research

The research and study of world-wide standards made by Ex Libris that also includes Ex Libris' practice and experience in the digital preservation arena.

The following is a summary of that research:

- Repository standards

OAIS – This is the ISO reference model for the Open Archival Information System. It was agreed, at an early stage of the project, that the preservation system designed and developed for the project would be OAIS compliant.

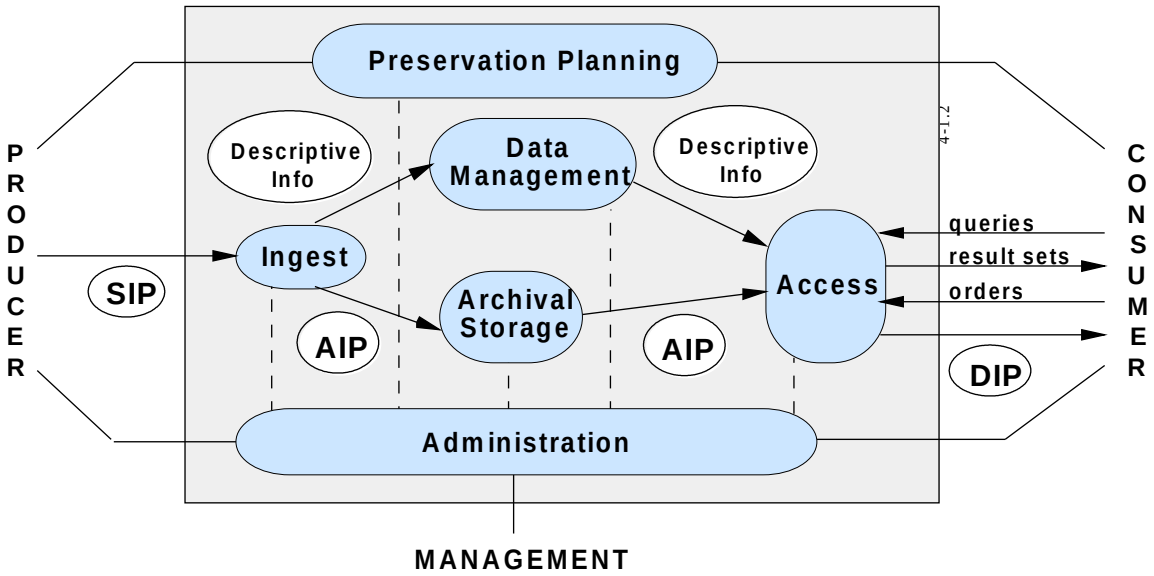


Figure 1: OAIS Functional Entities

TDR – This defines the guidelines, roles, and responsibilities for a digital repository in order for it to be considered trusted.

TRAC – This provides tools for the audit, assessment, and certification of digital repositories.

- Metadata standards

PREMIS – This defines the metadata strategies and data model for use in digital preservation.

METS – This is a standard for encoding the descriptive, administrative, and structural metadata regarding objects within a digital library.

Dublin Core – This defines the metadata elements to describe digital objects for collections management and for the exchange of metadata.

- Exchange standards

  The **SRU/SRW** and the **OAI-PMH.**

- Tools/technologies

  While reviewing different tools, parameters like ease of use, frequency of up-dates, sustainability, and so on were taken into account. Knowing that over the long-term those tools will be changed and that new and better tools will be de-veloped, the preservation system should be open and capable to enable tools to be plugged in easily.

  The following are the categories evaluated:

  - File identification tools
  - Technical MD extractors
  - Fixity tools
  - Virus check
  - Workflow engines

- Design

The design of a preservation system should take into account all of the issues/topics mentioned above and be described in a detailed document.

- Specification

The production of a specification document with a detailed description for how each of the selected/chosen standards and tools should be implemented within a preservation system.

Under the digital preservation component, the work done includes:

- Reviews of organizations/projects dealing with the long-term preservation of audiovisual content

Several of the leading projects/organizations dealing with the preservation of audi-ovisual content were evaluated. The organizations and projects that were evaluated include The Association of Moving Image Archivists, the Association for Recorded Sound Collections, the IASA, the Bay Area Video Coalition, the British Universities Film & Video Council, The Educational Recording Agency (ERA), PrestoSpace, and the Federal Agencies Digitization Guidelines.

The conclusion of this survey is that, as of today (late 2009), none of the above have either completed the research required or have published a comprehensive definition and/or guideline for the digital preservation of audiovisual content.

- Preservation framework

According to OAIS, preservation is a component on top of the digital repository that manages the preservation action. Design of a preservation component should provide a framework for identifying and reporting risks, defining several alternatives, running alternatives in a test mode, automatic and manual evaluation of test results, and the sign off and execution of a plan.

1 DPS Development Background

A key driver in the design of a digital preservation system (DPS) for AV materials was to ensure that the system would be widely applicable. The research in this chapter focused on understanding the tools and standards most recommended not only from an academic or theoretical point of view but also from a “best practices” point of view.

1.1 International Standards

A guideline for designing and developing a DPS was to adhere to international standards. As DPS will be increasingly in use, it is crucial for it to adopt international standards in all aspects of the system – business model, metadata, tools, and so on.

Following are the standards adopted by the Ex Libris DPS.

1.1.1 General

1.1.1.1 OAIS

Description:

OAIS is the ISO reference model for Open Archival Information System. The OAIS reference model is defined by a recommendation of the Consultative Committee for Space Data Systems.

Use in Rosetta:

The OAIS model is used in the Rosetta system across all its modules. All of Rosetta’s key components are aligned with the OAIS model (ingest, archival storage, data management, access, and preservation planning). In addition, the data flow and data model in the Rosetta system matches those described in the OAIS reference model document.

For example, when depositing material into the system, the data is stored in a submission information package (SIP).

For further information:

- http://public.ccsds.org/publications/archive/650x0b1.pdf
- http://en.wikipedia.org/wiki/Open_Archival_Information_System

1.1.1.2 TDR/TRAC

Description:

Trusted Digital Repository (TDR) was published in May 2002 by RLG[1] and OCLC[2]. TDR defines guidelines, roles, and responsibilities for a digital repository in order for it to be considered trusted.

[1] currently (2009) part of the OCLC research group
[2] http://www.oclc.org/default.htm

Trustworthy Repositories Audit & Certification (TRAC) was published in May 2007 after three years of work done by DCC[3], RLG, NARA[4], nestor[5], and the US Center for Research Libraries[6]. The end result of the international collaboration is a guideline that:

- Provides tools for the audit, assessment, and potential certification of digital repositories
- Establishes documentation requirements necessary for audits
- Delineates a process for certification
- Establishes appropriate methodologies for determining the soundness and sustainability of digital repositories

Use in Rosetta:

Rosetta was developed based on TDR and TRAC guidelines and is compliant with both.

For further information:

- http://www.oclc.org/research/activities/past/rlg/trustedrep/default.htm
- http://www.dcc.ac.uk/tools/trustworthy-repositories/

1.1.2 Metadata

1.1.2.1 PREMIS

Description:

PREservation Metadata: Implementation Strategies (PREMIS[7]) is an international working group concerned with developing metadata for use in digital preservation. The PREMIS data model consists of five interrelated entities: Intellectual, Object, Event, Agent, and Rights with each semantic unit mapped to one of these areas.

Use in Rosetta:

Rosetta uses the PREMIS data model as its core model. When materials are deposited in the system, they are immediately transformed into this model of intellectual entities, representations, and files, each level with its own relevant metadata. For example, a book will be kept as a single intellectual entity. Each representation will hold a copy of the book (original copy, access copy, master copy). Each representation will hold the pages of the book.

In addition, Rosetta uses the PREMIS model for provenance events.

For further information:

- http://en.wikipedia.org/wiki/PREMIS
- http://www.loc.gov/standards/premis/v2/premis-2-0.pdf

[3] http://www.dcc.ac.uk/
[4] http://www.archives.gov/
[5] http://www.langzeitarchivierung.de/eng/index.htm
[6] http://www.crl.edu/
[7] http://www.loc.gov/standards/premis/

1.1.2.2 METS

Description:

The Metadata Encoding and Transmission Standard (METS[8]) schema is a standard for encoding descriptive, administrative, and structural metadata regarding objects within a digital library. METS uses the XML schema language of the World Wide Web Consortium.

Use in Rosetta:

The METS schema is used in the Rosetta system as the container of the Intellectual Entity. It stores the data model which is compliant with the PREMIS structural model and its descriptive and administrative metadata. A Rosetta METS file holds the intellectual entity, its representations and files, and the metadata for each of these items.

For example, when material is deposited in the system, a METS file is created to represent the OAIS SIP. The file will hold the deposited material structure and its metadata according to the PREMIS data model.

For further information:

- http://en.wikipedia.org/wiki/METS
- http://www.loc.gov/standards/mets
- http://www.loc.gov/standards/mets/mets.xsd (latest METS schema)

1.1.2.3 Dublin Core

Description:

Dublin core (DC) is a simple set of metadata elements used in digital libraries, primarily to describe digital objects and for collections management, and for exchange of metadata.

Use in Rosetta:

The Rosetta system uses DC for descriptive metadata.

For further information:

- http://dublincore.org
- http://dublincore.org/documents/abstract-model
- http://dublincore.org/schemas/xmls

1.1.3 Infrastructure

1.1.3.1 Java 5

Use in Rosetta:

The Rosetta system is written in Java, version 5.

For further information:

[8] http://www.loc.gov/standards/mets/

- http://www.sitepoint.com/article/introducing-java-5/
- http://java.sun.com/j2se/1.5.0/

1.1.3.2 JBOSS

Description:

JBoss Application Server (JBoss AS) is an open-source[9], Java EE[10]-based application server[11] that operates across all platforms. It is usable on any operating system that Java supports.

Use in Rosetta:

The Rosetta's Web service is JBoss 4.2 (may upgrade)

For further information:

- http://jboss.org/jbossas
- http://en.wikipedia.org/wiki/Jboss

1.1.3.3 JBPM

Description:

jBPM is a platform for executable process languages ranging from business process management (BPM) over workflow to service orchestration.

Use in Rosetta:

The Rosetta's web service is JBoss 4.2 (may upgrade)

For further information:

- http://jboss.org/jbossjbpm

1.1.3.4 Tikal Eclipse

Description:

Tikal Eclipse is an open-source Eclipse-based distro which combines the Eclipse binaries with a broad range of Eclipse plug-ins for Java, Java Enterprise, C/C++ and Dynamic Languages (Perl, PHP, Python and Ruby) Development. Tikal Eclipse has a fast and easy graphical installer. We have kept the distribution installation binaries set down to one small single file. Everything else you might need is available online.

Use in Rosetta:

Ex Libris' Rosetta developers use the Tikal eclipse.

[9] http://en.wikipedia.org/wiki/Open-source_software
[10] http://en.wikipedia.org/wiki/Java_EE
[11] http://en.wikipedia.org/wiki/Application_server

For further information:

- http://tikal.sourceforge.net/tikal-eclipse/index.html

1.1.4 Exchange

1.1.4.1 SRU / SRW

Description:

Search/Retrieve Web service (SRW) provides a SOAP interface to queries that augments the URL interface provided by its companion protocol, Search/Retrieve via URL (SRU). Queries in SRU and SRW are expressed using the Common Query Language (CQL).

Use in Rosetta:

Rosetta adopted these standards for Web service (either SOAP or URL) for searching entities and metadata and for all other common services. These Web services are for external use but are also used frequently across the system for internal purposes.

For further information:

- http://en.wikipedia.org/wiki/SOAP/http://en.wikipedia.org/wiki/Web_Service
- http://www.loc.gov/standards/sru
- http://www.loc.gov/standards/sru/sru1-1archive/xml-files.html

1.1.4.2 OAI-PMH

Description:

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a protocol developed by the Open Archives Initiative. It is used to harvest (or collect) the metadata descriptions of the records in an archive so that services can be built using metadata from many archives. An implementation of OAI-PMH must support representing metadata in Dublin Core, but may also support additional representations.

Use in Rosetta:

Rosetta uses the OAI-PMH protocol as part of its Publishing module.

For further information:

- http://www.openarchives.org/

1.2 Peer Review Group

An international Peer Review Group (PRG) has been formed as an independent resource for the NDHA Program.

The PRG consists of recognized leaders in thought and innovation from the international library and academic communities. All have institutional expertise in the areas of digital preservation and permanent access.

Group members are from the following institutions:

- The British Library
- Cornell University Library
- The Getty Research Institute
- The National Library of Finland
- The National Library of New Zealand
- The National Library of the Netherlands
- The National Library of China
- National Library Singapore
- The University of Glasgow
- Yale University

1.3 Related Projects

Long-term preservation of digital objects is an emerging challenge for many institutions world-wide. Ex Libris has been considering the following projects and initiatives while developing DPS:

- KOPAL - http://www.d-nb.de/eng/wir/projekte/kopal.htm
- DAITSS - http://www.fcla.edu/digitalArchive/pdfs/DAITSS.pdf
- E-Depot - http://www.kb.nl/dnp/e-depot/operational/background/index-en.html
- NLNZ preservation metadata standards - http://www.natlib.govt.nz/catalogues/library-documents/preservation-metadata-revised
- SPAR (a project at the BNF) - http://www.britannica.com/bps/additional-content/18/34229915/Preserving-Digital-Information-at-the-BnF-The-SPAR-Project
- Fedora commons - http://www.fedora-commons.org/
- LOCKSS - http://www.lockss.org/lockss/Home
- CASPAR - http://www.casparpreserves.eu/
- PLANETS - http://www.planets-project.eu/
- SHAMAN - http://shaman-ip.eu/shaman/

2 Review of Current Digital Preservation Support for AV

A review of current digital preservation support for AV should include two aspects:

- Business aspect - Review of several organizations currently dealing with digital preservation for AV, what has been published to-date and what is currently being planned. This will be covered in this section.
- Technical aspect – What tools are available such as technical MD extraction, fixity, virus check. This will be covered in chapter 4 - Tools assessment.

2.1 The Association of Moving Image Archivists[12]

The Association of Moving Image Archivists (AMIA) is a non-profit professional association established to advance the field of moving image archiving by fostering cooperation among individuals and organizations concerned with the acquisition, description, preservation, exhibition, and use of moving image materials.

2.2 Association for Recorded Sound Collections (ARSC)[13]

The Association for Recorded Sound Collections (ARSC) is an organization dedicated to the preservation and study of sound recordings, in all genres of music and speech, in all formats, and from all periods. ARSC is unique in bringing together private individuals and institutional professionals with a serious interest in recorded sound.

2.3 IASA[14]

The International Association of Sound and Audiovisual Archives (IASA) was established in 1969 in Amsterdam to function as a medium for international cooperation between archives that preserve recorded sound and audiovisual documents.

IASA practices the dissemination of information on collection development and access, documentation and metadata, copyright and ethics, conservation and preservation.

IASA follows closely the progress of technology, and the association's members can call on a pool of expert advisors for help with digitization and with problems arising from the use of computer storage systems for heritage collections.

2.4 Bay Area Video Coalition[15]

BAVC works to preserve and digitize precious works of media art and other cultural artefacts. Since 1994, BAVC has preserved over 6,000 hours of audio and videotape.

[12] http://www.amianet.org/
[13] http://www.arsc-audio.org/
[14] http://www.iasa-web.org/
[15] http://www.bavc.org/

2.5 British Universities Film & Video Council[16]

The Film Archive Forum represents all of the public sector film and television archives that preserve the UK's moving image heritage. The Forum represents all archival aspects of the moving image, and acts as the advisory body on national moving image archive policy.

2.6 The Educational Recording Agency (ERA)[17]

On behalf of its members, ERA operates a Licensing Scheme for educational use of copyright material. Uniquely serving the UK educational sector, ERA is one of a range of collecting societies that help copyright owners and performers derive an income from the licensed use of their literary, dramatic, musical, and artistic works. The ERA License relates to recording of both radio and television broadcasts.

2.7 PrestoSpace[18]

Institutions traditionally responsible for preserving audiovisual collections (broadcasters, research institutions, libraries, museums) now face major technical, organizational, resource, and legal challenges in taking on the migration to digital formats and the preservation of already digitized holdings. Technical obsolescence and physical deterioration of assets indicate the need for a widely employed policy and efficient technical services to achieve long-term digital preservation. The principal aim is to build up preservation factories providing affordable services to the custodians of these collections in order to allow them to manage and distribute their assets.

2.8 Federal Agencies Digitization Guidelines[19]

The goal of this project is to identify, establish, and disseminate information about standards and practices for the digital reformatting of audiovisual materials by federal agencies. The acceptance of a common body of digitization standards and practices will

- provide the public with products of uniform quality
- set common benchmarks for digitization service providers
- support content preservation for the long term, and
- facilitate the exchange of findings from related research.

The effort will cover sound and video recordings and motion picture film. The main focus of the work is on the reformatting of older materials, with limited discussion of the formatting of born-digital content. The term formatting is used here to cover

a. encoding
b. file formats and/or wrappers, for example, the Broadcast Wave File and the Material Exchange Format (MXF) and
c. content-bundling formats, ranging from examples like METS to ZIP.

[16] http://bufvc.ac.uk/
[17] http://www.era.org.uk/
[18] http://prestospace.org/
[19] http://www.digitizationguidelines.gov/

The activity will also examine the non-descriptive metadata associated with the content and the digitization process, metadata sometimes called technical, administrative, digital provenance, and preservation. The project will consider metadata encapsulated within files or other packages as well as metadata maintained or transmitted in conjunction with files. Although derivative representations (for example, listening or viewing versions for Web dissemination) of content are within scope, the primary focus of this project is the creation and validation of digital masters.

2.9 Summary

Of the many projects/initiatives around the world dealing with digital preservation with special attention to AV, only three projects/organizations have actually started defining guidelines for digital preservation for AV:

- IASA started defining guidelines for digital preservation for AV (available on **IASA TC04**, Guidelines on the Production and Preservation of Digital Audio Objects), but they do not intend to preserve their own collections
- Federal Agencies have established a working group dealing with AV aspects
- PrestoSpace was involved in digitisation, restoration, automatic metadata extraction and was not focused on long term preservation

As of today (late 2009), none of the above has either completed the research required or have published a comprehensive definition/guideline for digital preservation for AV.

3 Tools Assessment

A review of current digital preservation support for AV should include the technical aspects, such as what tools are available. For each category, such as MD extraction, fixity, virus check, and so on, a thorough search and review was performed on several prominent and leading tools, with the tool best suited for the task selected.

3.1 Technical MD Extractor

Metadata extraction will be performed on all files ingested into the repository as part of the ingest workflow. This metadata consists of basic file information (such as name and size) and mime-type specific information, such as bit rate, sample-rate, resolution, etc.

The ideal product would:

- Be able to extract metadata from a wide variety of file formats
- Be easily updatable to allow for new/updated file formats
- Be easily integrated (via Java or command-line interface)
- Have ongoing development/community
- Have a small footprint

As part of the research done for a DPS for AV, the following utilities were analyzed:

- libextractor
- NLNZ Metadata Extractor
- Soft Experience Metadata Miner Catalogue PRO
- Xena
- exif
- JHOVE
- alphaWorks

3.1.1 Libextractor

http://gnunet.org/libextractor/

Background:

libextractor is a library used to extract metadata from files of arbitrary type. It is designed to use helper-libraries to perform the actual extraction, and to be trivially extendable by linking against external extractors for additional file types. Libextractor is part of the Gnu project.

Currently (2009), libextractor supports the following formats: HTML, PDF, PS, OLE2 (DOC, XLS, PPT), OpenOffice (sxw), StarOffice (sdw), DVI, MAN, FLAC, MP3 (ID3v1 and ID3v2), NSF(E) (NES music), SID (C64 music), OGG, WAV, EXIV2, JPEG, GIF, PNG, TIFF, DEB, RPM, TAR(.GZ), ZIP, ELF, S3M (Scream Tracker 3), XM (eXtended Module), IT (Impulse Tracker), FLV, REAL, RIFF (AVI), MPEG, QT and ASF.

Also, various additional MIME types are detected.

The main libextractor module itself basically exists to manage the extraction – the actual extraction is handled by the helper libraries, which libextractor invokes based on the mime type of the file being processed.

Advantages:

- Ongoing development.
- Free – open source and easily extendible.
- Supports wide variety of file types.
- There is a java interface to libextractor available using JNI. Also command-line support via Extract program
- Also computes checksums (MD5, for example)

Disadvantages:

- It does not seem to retrieve any basic file metadata – it retrieves mime-type specific information only.
- Written in C and requires a number of (open-source) C libraries.

3.1.2 NLNZ Metadata Extractor v1.0

http://www.natlib.govt.nz/en/whatsnew/4initiatives.html#extraction

Background:

The preservation metadata extraction tool:

a) automatically extracts preservation-related metadata from digital files, and

b) outputs that metadata in a standard format (XML) for uploading into a preservation metadata repository.

The Java/XML tool comprises a generic application and a number of “adapters” developed to extract the data from specific file types. To-date, adapters have been written for MS Word 2, MS Word 6, Word Perfect, Open Office, MS Works, MS Excel, MS PowerPoint, TIFF, JPEG, WAV, MP3, HTML, PDF, GIF, and BMP

The extractor has a GUI but can also be accessed through the command line. The adapters are ‘installed’ into the main program and the file(s) are then harvested for their metadata. Adapters are all separate JARs, making integrating new adapters or updating existing ones simple.

Advantages:

- Java – easy integration with DPS
- Free – Open source.
- Metadata output format is XML and can be configured easily.
- Extraction adapters can be swapped in/out easily.
- Small footprint.
- Supports most popular file types (except ZIP, TAR).

Disadvantages:

- Doesn't support MP3 ID3V2 and partial support of RIFF(wav) files.
- No ongoing support.

3.1.3 Soft Experience Metadata Miner Catalogue PRO

http://peccatte.karefil.com/software/Catalogue/MetadataMiner.htm#Batch_PRO

Background:

Listing folders and documents summary information, extraction of file properties and metadata capture, creation of HTML file catalogs including hypertext links and building XML reports.

Software for extended listings of folder contents in interactive mode and in command line. The file cataloguing utility enables quick creation of HTML pages listing files and associated metadata, and managing and updating document properties associated with such files as: Microsoft Office, retrieve information from OpenOffice.org, StarOffice, Visio documents, PDF documents, export metadata from photos JPEG, Tiff images (IPTC fields).

Metadata Miner Catalogue PRO also enables Adobe XMP file information (eXtensible Metadata Platform) metadata extraction from documents produced by recent Adobe applications and XSL transformation on metadata extracted listings converting to Dublin Core RDF, CSV, SVG and other custom presentations for common uses or in a Content Management System Architecture.

Advantages:

- Supports most popular file types (except ZIP, TAR)
- Ongoing support
- Has command-line interface.

Disadvantages:

- Poor support for Multimedia.
- Runs on Windows machines only.
- Not free
- Probably more capability than needed.

3.1.4 exif

http://drewnoakes.com/code/exif/

Background:

Created initially to extract metadata from JPGs only, it has since been expanded to other file types using EXIF and IPTC formats.

Not seriously considered because the number of supported file formats is severely limited.

3.1.5 JHOVE

http://hul.harvard.edu/jhove/

Background:

JHOVE provides functions to perform format-specific identification, validation, and characterization of digital objects.

- Format identification is the process of determining the format to which a digital object conforms
- Format validation is the process of determining the level of compliance of a digital object to the specification for its purported format
- Format characterization is the process of determining the significant format-specific properties of an object of a given format.

The set of characteristics reported by JHOVE about a digital object is known as the object's representation information, a concept introduced by the OAIS reference model (ISO/IEC 14721). The standard representation information reported by JHOVE includes: file pathname or URI, last modification date, byte size, format, format version, MIME type, format profiles, and optionally, CRC32, MD5, and SHA-1 checksums (CRC32, MD5, SHA-1). Additional media type-specific representation information is consistent with the NISO Z39.87 Data Dictionary for digital still images and the draft AES metadata standard for digital audio.

For the purposes of metadata extraction, we are most interested in the last bullet, as it is the characterization feature that retrieves the metadata information about a given file.

Advantages:

- Will also perform format identification/validation duties that are needed by DPS
- Written in Java – easy integration
- Active community/development

Disadvantages:

- Limited number of file types (not ZIP, TAR, OLE2, open office, etc.).

3.1.6 alphaWorks

http://www.alphaworks.ibm.com/tech/webextractor

Background:

Web Metadata Extractor extracts relevant information from well-formed Web pages in HTML format by using semi-automatically generated templates based on the Unstructured Information Management Architecture (UIMA) framework. This information consists of metadata of the entity described in the Web page, such as title, author, source.

Not seriously considered because the number of supported file formats is severely limited (HTML only).

3.1.7 Summary

There is no perfect answer here.

Libextractor has the most comprehensive list of supported formats, but it doesn't retrieve general file technical metadata. One possibility would be to use the NLNZ tool to retrieve file technical metadata and to use libextractor to retrieve mime-specific information. The drawback would be the added complexity of this implementation and the added overhead due to the build complexity of the Gnu C libraries in addition to the Java build.

The NLNZ tool offers easy integration and a sufficient list of supported file types. It is not currently supported, however, so updates would be the responsibility of Ex Libris.

JHOVE is an option. For the file types it handles, it retrieves all file information (both file and mime-specific metadata). For the file types it doesn't support, it will verify that the file is valid and return limited metadata. It could be used to perform format identification/validation in addition to metadata extraction. The significant disadvantage with JHOVE is the relatively small number of file types it supports. There is no clear projection for when the number of supported file types will be expanded. Using JHOVE would probably result in considerable preservation work as file types previously unsupported become supported.

The other products do not meet the needs of DPS.

3.2 File Identification

File format identifiers attempt to determine the format of a given file and whether or not that file is a valid example of that format. While other utilities such as metadata extractors need to know the format of a file to extract the metadata, they often do not perform an in-depth analysis to determine the format. Instead, these utilities often rely on such basic information as the file extension and perhaps a mime-type descriptor byte.

File format identifiers generally try to look at the interior structure of the file, such as the header structure and perhaps other elements throughout the file.

The ideal product would:

- Support a large number of file formats
- Allow easy updating of file format information
- Perform in-depth file analysis
- Have ongoing support to update file format information
- Allow easy integration
- Have a small footprint

The utilities analyzed here are:

- JHOVE
- DROID

3.2.1 JHOVE

http://hul.harvard.edu/jhove/

Background:

JHOVE, introduced earlier in this document as it relates to metadata extraction (see JHOVE), provides functions to perform format-specific identification, validation, and characterization of digital objects.

In addition, JHOVE retrieves the metadata describing the file, such as file pathname, mime type, size and other metadata as appropriate. It can also create checksums for the file.

In the process of format identification/validation, files are passed to each processing module, and the validation module attempts to validate the file. While this design allows JHOVE to use in-depth validation methods (validating the full contents of the file as opposed to just the header), it makes updating existing formats or adding new file formats difficult.

Advantages:

- Can also be used to retrieve metadata and to create/verify checksums
- Utilizes in-depth methods of validation (whole file)
- Easily integrated

Disadvantages:

- Limited number of formats
- Difficult to update or add new formats

3.2.2 DROID

http://en.wikipedia.org/wiki/PRONOM_technical_registry

Background:

DROID (Digital Record Object Identification) is a software tool developed by The National Archives to perform automated batch identification of file formats. Developed by its Digital Preservation Department as part of its broader digital preservation activities, DROID is designed to meet the fundamental requirement of any digital repository: to be able to identify the precise format of all stored digital objects, and to link that identification to a central registry of technical information about that format and its dependencies.

DROID uses internal and external signatures to identify and report the specific file format versions of digital files. These signatures are stored in an XML signature file, generated from information recorded in the PRONOM technical registry. New and updated signatures are regularly added to PRONOM, and DROID can be configured to automatically download updated signature files from the PRONOM website via web services. DROID is a platform-independent Java application, and includes a documented, public API, for ease of integration with other systems. It can be invoked from two interfaces:

- A Java Swing GUI

- A command line interface

DROID allows files and folders to be selected from a file system for identification. This file list can be saved at any point. DROID can also be used to identify URIs and streams (command line interface only). After the identification process had been run, the results can be output in XML, CSV, or printer-friendly formats.

DROID contains information about a large variety of file formats (currently over 700), though a significant amount of this format information consists of only file extensions. For some of the more common file types, the format information is fairly specific and in-depth, such as checking header structure.

Advantages:

- Large number of formats
- Format information easily updateable
- Easy integration with Java
- Active community for updates to format information and development

Disadvantages

- The quality of the signature information varies from format to format
- File format validation algorithms generally not as in-depth as JHOVE

3.2.3 Summary

Because they are Java, both products can be integrated easily. Both products have a comparable footprint and both are open source, with ongoing development.

While DROID's validation algorithm is not as in-depth as JHOVE's, it is adequate for most formats and they are working to improve them. DROID's advantage is that it links to the PRONOM registry for format information, allowing it access to a large number of supported formats. The PRONOM registry benefits from the active development of new format information. The ability to easily update DROID with this information makes it more desirable than JHOVE.

3.3 Fixity Check

Some of the fixity check algorithms or hash algorithms that might be suitable for the DPS project are the cyclic redundancy check (CRC), Fletcher's checksum, Adler-32, Message-Digest 5 (MD5), Secure Hash Algorithm (SHA), and RACE Integrity Primitives Evaluation Message Digest (RIPEMD).

3.3.1 Cyclic Redundancy Check

The following information is based on a recent definition from Wikipedia (http://en.wikipedia.org/wiki/Cyclic_redundancy_check):

A cyclic redundancy check (CRC) is a type of hash function used to produce a checksum – a small, fixed number of bits – against a block of data, such as a packet of network traffic or a block of a computer file. The checksum is used to detect errors after

transmission or storage. A CRC is computed and appended before transmission or storage and verified afterwards by the recipient to confirm that no changes occurred in transit. CRCs are popular because they are simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels.

While useful for error detection, CRCs cannot be safely relied upon to verify data integrity (that no changes whatsoever have occurred) since, because of the linear structure of CRC polynomials, it is extremely easy to intentionally change data without modifying its CRC.

3.3.2 Fletcher's Checksum

The following information is based on a recent definition from Wikipedia:

Fletcher's checksum is one of several types of checksum algorithms, which are relatively simple processes used by computers to check the integrity of data. It is designed to overcome some of the inadequacies of simply summing all the bytes as in the original checksum. Fletcher's checksum, unlike the original checksum, can detect the inserting/deleting of zero value bytes, the reordering of bytes, and incrementing and decrementing of the bytes in opposite directions.

Fletcher's checksum is described in the Internet standards document Request for Comments (RFC) 1146. You can also find information about generating (as well as verifying) such a checksum in Annex B of RFC 905. The RFC states that there are two versions of the Fletcher algorithm, an 8-bit algorithm which gives a 16-bit checksum and a 16-bit algorithm which gives a 32-bit checksum.

3.3.3 Adler-32

The following information is based on a recent definition from Wikipedia: Adler-32 is a checksum algorithm which was invented by Mark Adler. It is almost as reliable as a 32-bit cyclic redundancy check for protecting against accidental modification of data, such as distortions occurring during a transmission, but is significantly faster to calculate in software.

Adler sums are computed over 8-bit bytes rather than 16-bit words (as in Fletcher's), resulting in twice the number of loop iterations. This results in the Adler-32 checksum taking between one-and-a-half to two times as long as Fletcher's checksum.

Adler-32 has the benefit over a CRC that it can be computed faster in software. But Adler-32 has a weakness for short messages with few hundred bytes, because the checksums for these messages have a poor coverage of the 32 available bits.

3.3.4 MD5

The following information is based on a recent definition from Wikipedia:

In cryptography, MD5 (Message-Digest algorithm 5) is a widely used cryptographic hash function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. However, it has been shown that MD5 is not collision resistant; as such, MD5 is not suitable for applications like SSL certificates or digital signatures that rely on this property. An MD5 hash is typically expressed as a 32-digit hexadecimal number.

3.3.5 SHA

The following information is based on a recent definition from Wikipedia:

The SHA (Secure Hash Algorithm) family is a set of related cryptographic hash functions. The most commonly used function in the family, SHA-1, is employed in a large variety of popular security applications and protocols, including TLS, SSL, PGP, SSH, S/MIME, and IPSec. SHA-1 is considered to be the successor to MD5, an earlier, widely-used hash function. Both are reportedly compromised. In some circles, it is suggested that SHA-256 or greater be used for critical technology. The SHA algorithms were designed by the National Security Agency (NSA) and published as a US government standard.

The first member of the family, published in 1993, is officially called SHA; however, it is often called SHA-0 to avoid confusion with its successors. Two years later, SHA-1, the first successor to SHA, was published. Both SHA-0 and SHA-1 produce a 160-bit digest from a message. Four more variants have since been issued with increased output ranges and a slightly different design: SHA-224, SHA-256, SHA-384, and SHA-512 — sometimes collectively referred to as SHA-2.

Attacks have been found for both SHA-0 and SHA-1. No attacks have yet been reported on the SHA-2 variants, but since they are similar to SHA-1, researchers are worried, and are developing candidates for a new, better hashing standard.

3.3.6 RIPEMD

The following information is based on a recent definition from Wikipedia:

RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest) is a 160-bit message digest algorithm (and cryptographic hash function) developed in Europe by Hans Dobbertin, Antoon Bosselaers and Bart Preneel, and first published in 1996. It is an improved version of RIPEMD, which in turn was based upon the design principles used in MD4, and is similar in performance to the more popular SHA-1.

There also exist 128, 256 and 320-bit versions of this algorithm, called RIPEMD-128, RIPEMD-256, and RIPEMD-320, respectively. The 128-bit version was intended only as a drop-in replacement for the original RIPEMD, which was also 128-bit, and which had been found to have questionable security. The 256 and 320-bit versions diminish only the chance of accidental collision, and don't have higher levels of security as compared to, respectively, RIPEMD-128 and RIPEMD-160.

RIPEMD-160 was designed in the open academic community, in contrast to the NSA-designed algorithm, SHA-1. On the other hand, RIPEMD-160 is a less popular and correspondingly less well-studied design.

3.3.7 Checksum vs. Secure Hash

We have reviewed several types of checksum algorithms and several types of secure hash algorithms.

A checksum is a form of redundancy check, a very simple measure for protecting the integrity of data by detecting errors in data that is sent through space (telecommunications) or time (storage). It works by adding up the basic components of a message,

typically the asserted bits, and storing the resulting value. Later, anyone can perform the same operation on the data, compare the result to the authentic checksum, and (assuming that the sums match) conclude that the message was probably not corrupted.

Most sophisticated types of redundancy check, including Fletcher's checksum, Adler-32, and cyclic redundancy checks (CRCs), consider not only the value of each byte but also its position. The cost of the ability to detect more types of errors is the increased complexity of computing the checksum.

An important application of secure hashes (message digest) is the verification of message integrity that is the determination of whether or not any changes have been made to a message (or a file).

Below are some representative computational processing costs of checksums vs. hashes. (The numbers below can vary based on the machine configuration.)

Algorithm	Relative Cost
32-bit Adler	1.00
32-bit CRC (IEEE)	1.46
64-bit CRC (ISO)	2.23
128-bit CRC	2.29
128-bit MD4	4.48
128-bit MD5	4.51
160-bit SHA-1	10.20

3.3.8 Summary

In order to have the most wide coverage, three fixity methods will be used and stored as part of the AIP – CRC, SHA and MD5.

3.4 Virus Checks

It was reviewed the virus scan application software from the following companies: McAfee, Symantec, Frisk Software International (F-PROT), Trend Micro, Clam Anti Virus (ClamAV) and Sophos.

McAfee, Symantec, Trend Micro, and ClamAV were suggested by Reed Elsevier Security.

3.4.1 McAfee

3.4.1.1 Background

The following information was extracted from Wikipedia.

McAfee Virus Scan is an anti-virus tool made by McAfee. Another version, known as Virus Scan Enterprise, has also been released for the corporate environment.

McAfee is available on Widows, Mac, Linux, and Solaris. Below is some information from the product sheet for LinuxShield. There is also a brief mention of the Solaris offering.

LinuxShield detects and removes viruses and other potentially unwanted software on
Linux-based systems. LinuxShield uses the powerful McAfee scanning engine — the
engine common to all our anti-virus products.

Although a few years ago, the Linux operating system was considered a secure environment, it is now seeing more occurrences of software specifically written to attack or exploit security weaknesses in Linux-based systems. Increasingly, Linux-based systems interact with Windows-based computers. Although viruses written to attack Windows-based systems do not directly attack Linux systems, a Linux
server can harbour these viruses, ready to infect any client that connects to it.

When installed on your Linux systems, LinuxShield provides protection against viruses, Trojan horses, and other types of potentially unwanted software.

LinuxShield scans files as they are opened and closed — a technique known as on-access scanning. LinuxShield also incorporates an on-demand scanner that enables
you to scan any directory or file in your host at any time.

When kept up-to-date with the latest virus-definition (DAT) files, LinuxShield is an effective part of network security. For optimal security, set up an anti-virus security policy for your network, incorporating as many protective measures as possible.

LinuxShield uses a Web browser interface. A large number of LinuxShield installations can be centrally controlled by ePolicy Orchestrator.

3.4.2 Symantec

3.4.2.1 Background

The following information was extracted from Wikipedia.

In recent years, Symantec has been primarily known for its Norton-branded anti-virus and utility software. The Symantec Security Response organization (formerly Symantec Antivirus Research Center) is one of the foremost anti-virus and computer security research groups in the industry.

Symantec Scan Engine is a product for enterprise customers. Scan Engine is a TCP/IP server application and communication Application Programming Interface (API) that provides virus scanning services. Designed specifically to protect traffic served through, or stored on, network infrastructure devices, it detects and protects against viruses, worms, and Trojan horses in all major file types, including mobile code and compressed file formats. It is available for Solaris systems.

3.4.2.2 Assessment

Advantages

- Available on Solaris
- Application programmer interface available
- Symantec is the market leader in virus scan software

3.4.3 Trend Micro

3.4.3.1 Background

Trend Micro provides a product that could be an option for filtering inbound requests sent to DPS. They have a product that recognizes viruses in transit (via FTP or HTTP), but not one that provides basic virus scanning of files at rest on a disk. This product alone may not be sufficient for the DPS requirements.

Web Security Suite provides the first line of defence against multiple Web-based threats blocking attacks at the gateway. It guards against viruses, spyware, grayware, and phishing. Built on Trend Micro's award-winning technology, the latest scan engine filters HTTP and FTP traffic with minimal impact on Internet performance.

3.4.3.2 Assessment

Advantages

- Among the market leaders in virus scan software

Disadvantages

- No product available to scan files at rest on disk

3.4.4 FRISK Software International

3.4.4.1 Background

The following information was obtained from Wikipedia.

FRISK Software International (FSI), is an Icelandic software company that develops F-PROT anti-virus and F-PROT AVES anti-virus and anti-spam service. Its name is derived from the initial letters of the personal name and patronymic of Friðrik Skúlason, its founder. The company was founded in 1993.

F-PROT Antivirus is sold in both home and corporate packages, of which there are editions for Windows, Linux, and BSD. There is also a DOS version for home users, and corporate versions exist for Microsoft Exchange, Solaris, and certain IBM eServers. F-PROT has been produced since 1989.

F-PROT AVES is for corporate users.

FRISK Software International should not be confused with F-Secure, another data security company.

3.4.5 Sophos

3.4.5.1 Background

From Wikipedia:

Sophos is a company that makes security software such as anti-virus, anti-spyware and anti-spam for desktops, e-mail servers, and other network gateways.

SophosLabs, the company's global network of threat analysis centers, is an anti-virus and computer security research group in the computer security industry.

Unlike other security companies, Sophos does not produce anti-virus and anti-spam solutions for home users, but instead has always focused on the business market. Sophos Anti-Virus is aimed primarily at corporate environments.

Sophos Anti-Virus is available for the latest versions of Microsoft Windows, GNU/Linux, Novell NetWare, Mac OS X, VMS, and OS/2. It also supports UNIX platforms such as FreeBSD, Solaris, AIX, HPUX, SCO Group Openserver/UNIX-ware and Tru64.

The SAV Interface software development toolkit allows software developers to integrate the Sophos virus detection engine in their applications in C programming language or C++.

3.4.5.2 Assessment

Advantages

- Company focused on virus scan for business users

Disadvantages

- Higher priced than other vendors products.

3.4.6 Clam Anti Virus

3.4.6.1 Background

From Wikipedia:

Clam Anti Virus (ClamAV) is a widely used free anti-virus software toolkit for Unix-like operating systems. It is mainly used with a mail exchange server as a server-side email virus scanner. ClamAV is open source software distributed under the terms of the GNU General Public License (GPL). Both ClamAV and its updates are made available free of charge.

ClamAV is generally configured to automatically update its list of virus definitions via the Internet.

ClamAV provides a command-line scanner with on-access scanning for Linux and FreeBSD systems.

3.4.6.2 Advantages

- Free open source software.

3.4.7 Summary

The top best of breed solutions are McAfee (used for the most part across Reed Elsevier), Symantec and a toss between Trend Micro and F-PROT. An open-source solution would be ClamAV, which LexisNexis Group also uses.

3.5 Workflow Engine

A workflow engine is a tool that allows multiple tasks to be linked together into a single business process. The linking is accomplished in a manner that does not require code to be recompiled, so that processes may be changed or new processes may be added to a running system. Using a workflow engine requires that tasks be developed in such a way as to make them available to the engine.

For DPS, a workflow engine is a candidate technology for use in defining ingest processing and preservation processing.

The area of workflow engines is currently highly fragmented. Definition of a business process can be done using one of a number of different standards. While there does seem to be some convergence on BPEL (Business Process Execution Language), also known as BPEL4WS (BPEL for Web Services) for process definition, the implementation of that definition is done in different ways by different vendors. Given the current state of the technology, it is not reasonable to consider a workflow engine a “pluggable” part of the DPS system. While the activities that are invoked by a workflow engine may be independent, reimplementation of workflow processes for a different engine will be a substantial project.

3.5.1 The Open Source Business Engine

This is an open source Java workflow engine that is fully J2EE compliant and supports a number of J2EE application servers, databases and operating system. It is based on the WfMC[20] and OMG[21] spec. The engine doesn't run on threads. It is the APIs and common objects that handle the flow. This ensures continuous workflow even after system crashes.

OBE is highly configurable and extensible, and many aspects can be customized. The runtime engine relies upon pluggable services to provide authentication, authorization, persistence, task assignment, inbound event handling and outbound integration capabilities. OBE provides a workflow lifecycle event notification framework to support integration with workflow enabled applications.

OBE supports automated, manual and mixed workflow processes, and has extensible work item allocation and activity completion algorithms. Activities are automated through an extensible system of Tool Agents, which enable the invocation of external logic defined in Java classes, EJBs, native executables, scripts in arbitrary scripting languages, Web Services, and so on. Human interactions are managed through work items, which can be purely manual or can provide the means to invoke the appropriate software tools. OBE provides a worklist API and worklist clients to manage work items.

3.5.2 Flux

3.5.2.1 Background

This is widely used Java workflow engine. It ameliorates the productivity through job scheduling, File Transfer, Workflow and business process management (BPM) engine. APIs are available for Java, J2EE, XML, and Web Services. Flux can be used from the user interfaces without programming.

[20]Workflow Management Coalition
[21]Object Management Group

3.5.2.2 Advantages

Flux runs entire workflows in the background via a workflow manager. Flux is responsible for creating, managing, and throttling those threads. Flux provides explicit support for creating and killing operating system processes for running scripts and executables, either on the same computer as the Flux workflow engine or on a remote computer. Script output is collected and made available to the workflow for further processing.

For detecting and processing files arriving from remote servers, Flux provides explicit, built-in support for remote file monitoring.

3.5.2.3 Disadvantages

Flux does not provide the ability to pack workflow definition, associated Java classes, and all associated jar files into a single file, which makes it handy to keep all associated components of a workflow together.

3.5.3 OpenWFE

Implemented in Java, this open source workflow engine offers a complete Business Process Management Suite. OpenWFE features worklist component for storing work items, an APRE component for implementing automated agents into the work flows and the web based flow designer Droflo. There are 4 component involved in the BPM engine, worklist, webclient and reactor (host of automatic agents). It has a python access library that enables interaction between python application / client and OpenWFE REST worklist.

3.5.4 XFlow

This business processes management and workflows engine is built on a J2EE platform. It assists in integrating processes across an enterprise.
It is designed for easy development, deployment and management standpoints.

XFlow is a pure J2EE platform for building, executing and managing business processes and workflows. It is a basis for building collaborative applications as well as integrating processes across an enterprise. XFlow has a small footprint but is extremely powerful. It is designed to be easy to use from the development, deployment and management standpoints.

XFlow runs within an EJB and servlet container. JBoss 4.0 (with bundled Tomcat) is the container used in this implementation. The architecture supports distributed and parallel processes within an organization's firewall as well as across organizational boundaries.

XFlow is designed for scalability, extensibility and ease of integration with security models and legacy processes. XFlow's service-oriented architecture supports both a simple Java API as well as a web-service (SOAP/HTTP) interface.

3.5.5 ActiveBPEL

http://www.activebpel.org

http://www.active-endpoints.com

3.5.5.1 Background

The ActiveBPEL Engine is an open source implementation of a BPEL 1.1 engine, written in Java. The ActiveBPEL Engine runs in any standard servlet container such as Apache Tomcat.

The ActiveBPEL Engine *does not* contain a visual designer that allows you to easily and quickly create BPEL orchestrations. It *only* supports the execution of previously-coded BPEL 1.1 processes.

The ActiveBPEL Engine is made available under the GNU General Public License (GPL).

3.5.5.2 Advantages

The design tool is a top-notch development environment for business processes, allowing for interactive testing and debugging.

The commercially licensed versions take advantage of all J2EE application server features, such as clustering, failover, load balancing, etc.

From the user forums, there appears to be a large and active user community. Active Endpoints staff is active in the forums.

Most engine configuration can be easily accomplished using a supplied web application.

Process versioning is supported in the licensed implementations, allowing for a workflow to be altered without changing the execution of already existing instances of that flow.

3.5.5.3 Disadvantages

The ActiveBPEL engine is fully compliant with the BPEL standard. That means it has all the disadvantages of BPEL (see Summary section below) with no clear way of working around them.

Not all engine configuration tasks can be accomplished using a supplied web application. Some features require editing XML files, which are also altered by the web application. This mixed mode configuration has the potential to be mismanaged.

3.5.6 JBoss jBPM

http://www.jboss.com/products/jbpm

3.5.6.1 Background

JBoss jBPM enables the creation of business processes that coordinate between people, applications and services. It internally uses an engine that allows different process definition languages to be used. Its primary process definition language is jPDL (a variant of the standard XPDL). An advantage of jPDL over BPEL is that the language standard allows for tasks that are expected to have human interaction, which includes tasks that will be implemented in DPS.

Support services are available from JBoss. There are active user forums, wiki pages, and an open defect tracking system.

The base JBoss jBPM is deployed as a single jar file, which can be included in a simple standalone program, with almost no additional dependencies. Adding database persistence requires Hibernate, and the BPEL extension requires deployment under a servlet container or J2EE server, along with a full web services stack. (See discussion of BPEL in the summary below.)

JBoss jBPM delegates issues such as clustering, failover, load balancing, etc. to the deployment environment, which could be Tomcat, JBoss Application Server, any other J2EE application server, or a standalone program.

A design tool is available, built on the Eclipse platform, which allows business processes to be developed in a graphic environment. It automates much of the work involved in creating a “process archive”, the deployable unit that defines a workflow process.

3.5.6.2 Advantages

The jPDL programming model is much simpler than BPEL, while not losing the functionality needed for DPS. (See discussion of BPEL in the summary below.)

The design tool is pretty good.

Process versioning is supported, allowing for a workflow to be altered without changing the execution of already existing instances of that flow.

No licensing cost.

Good user guide, a lot of sample code that is Eclipse ready.

3.5.6.3 Disadvantages

jPDL appears to be a jBPM specific dialect of XPDL, an older workflow specification language. This disadvantage may be mitigated by BPEL support, or may not matter for DPS.

3.6 Summary

3.6.1 The Problem with BPEL

BPEL appears to be the emerging standard in common use for business process definition, and is the standard that most vendors seem to be talking about. However, BPEL is primarily intended for orchestration of web services provided by a heterogeneous collection of providers. To accomplish that goal, it relies on Web Services standards to shield the orchestration engine from any knowledge of the details of the services being orchestrated. That adds a lot of overhead in development, and at runtime. It also adds a lot of conceptual complexity in understanding the roles that tasks are assigned, and the messaging that passes between them. Defining a business process in BPEL will require a high degree of technical knowledge and skill, and a deep understanding of the BPEL language, even with the support of a first class tool like ActiveBPEL Developer.

The problem that needs to be solved for Rosetta is different. In Rosetta, a group of tasks may be selected and strung together to create a workflow or process. The Rosetta architecture and design expects that the tasks will not be isolated from each other and the rest of the Rosetta system to the extent BPEL implies. For example, they are expected them to share a single event notification system.

3.6.2 Recommendation

Given the problems with BPEL described above, there does not seem to be sufficient value added to Rosetta to make implementation of a BPEL compliant workflow engine worth the cost in time, effort, and complexity.

JBoss jBPM appears to be a stable tool with a large user base, flexible deployment options, and a programming model that makes sense. The effort and knowledge required to define a business process in jPDL is still large, but should be easier to gain than in BPEL.

It is worth noting that nothing in jPDL prevents a task executed under its control from making a call to an external web service. So if there is a need to link a task from an external provider into a workflow, the capability is available.

JBoss jBPM is recommended as a workflow engine.

4 System Design

4.1 OAIS Model – Overview

The OAIS model is shown in Figure 1.

4.1.1 From the OAIS document

The OAIS reference model provides:

- a framework for the understanding and increased awareness of archival concepts needed for Long Term digital information preservation and access
- the concepts needed by non-archival organizations to be effective participants in the preservation process
- a framework, including terminology and concepts, for describing and comparing architectures and operations of existing and future archives
- a framework for describing and comparing different long term preservation strategies and techniques
- a basis for comparing the data models of digital information preserved by archives and for discussing how data models and the underlying information may change over time
- a foundation that may be expanded by other efforts to cover long-term preservation of information that is NOT in digital form (e.g., physical media and physical samples)
- an expanded consensus on the elements and processes for long-term digital information preservation and access, and the promotion of a larger market which vendors can support
- a guide for the identification and production of OAIS-related standards

4.2 Rosetta overview

The following is a high-level description of Rosetta.

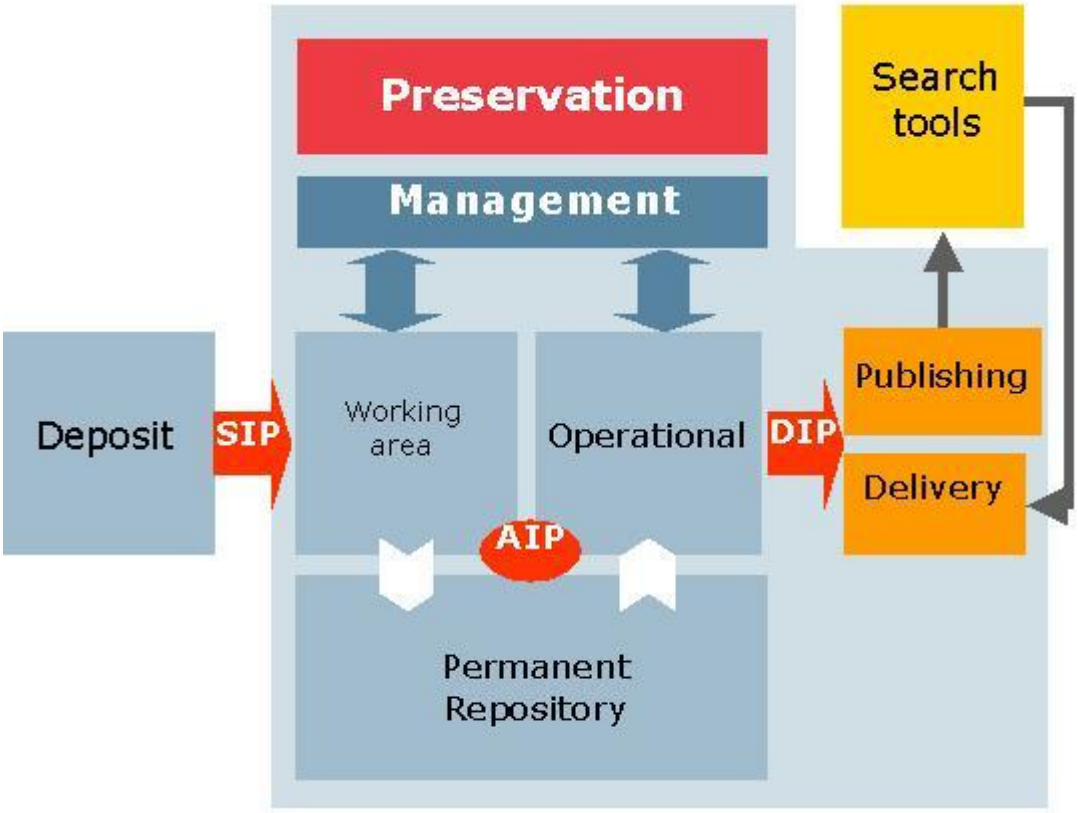


Figure 2: Rosetta components

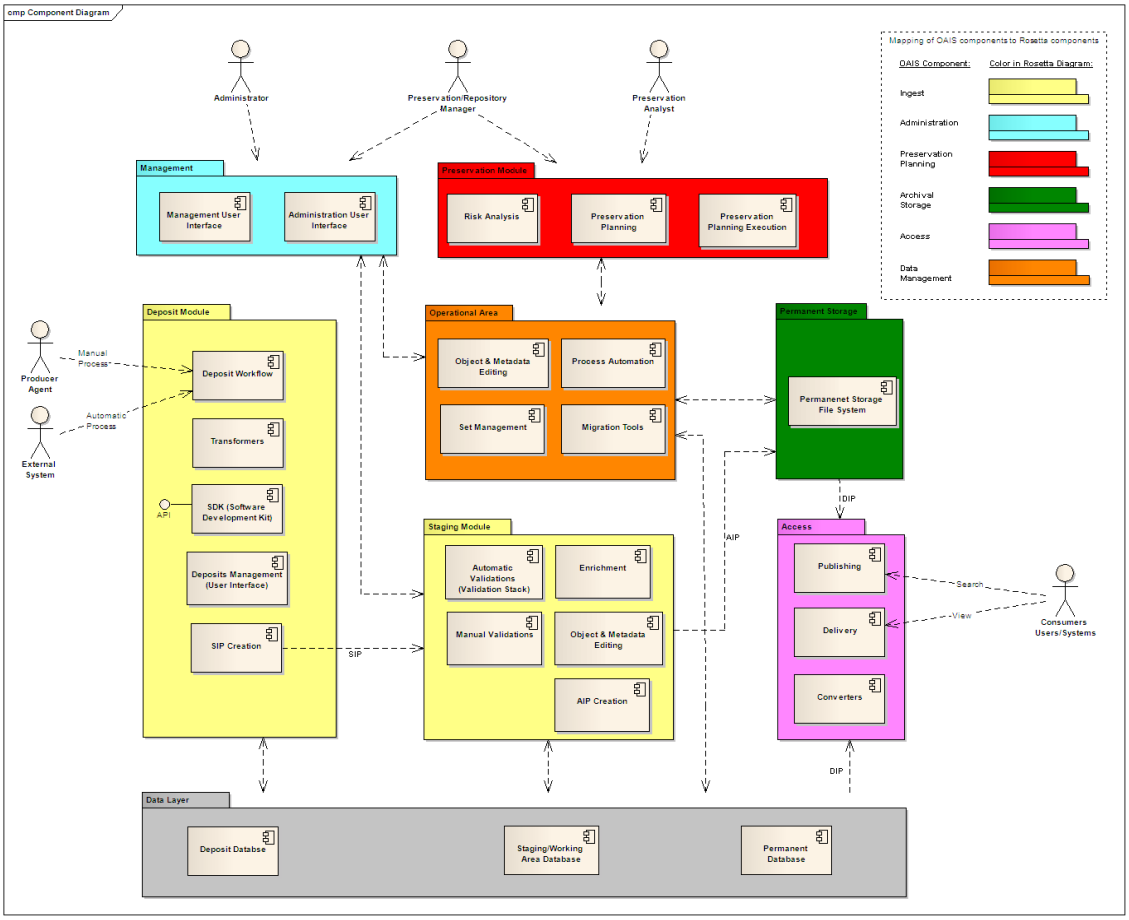


Figure 3: Rosetta software components

4.2.1 Deposit

Deposit module is the module that gets all the materials to be deposited into Rosetta as input and transforms it to a SIP structure.

Deposit module enables 3 different ways of depositing materials into Rosetta:

- Rosetta UI for loading file/s from user's PC
- Rosetta UI for loading files from a FTP location
- Integration with external submission applications (using Deposit API)

4.2.2 Working Area

Working area module is composed of the following:

- Loading – Loads information received in SIP structure into set of ORACLE tables in the repository database
- Validation stack is composed of the following:
  - File identification – Rosetta is using DROID as the file identification tool. The output of DROID is captured and stored as part of the AIP
  - Fixity – Rosetta uses 3 fixity methods: MD5, CRC32 and SHA1. The output of those 3 is captured and stored as part of the AIP
  - Virus check – Rosetta default virus check tool is a tool developed by NLNZ
  - Technical metadata extractor – Any technical metadata extractor tool (e.g., JHOVE, NLNZ Extractor) can be plugged in. The output of the MD extractor is captured and stored as part of the AIP
- Approval – Rosetta is configurable in a way that each material flow may require human interfere. In such case Rosetta staff member can: Assess the SIP (based on viewing the files and the metadata), Arrange it in different structure and finally approve it to be moved to permanent
- Enrichment – Adding external data such as connection to a Collection Management System (CMS) MD info

4.2.3 Permanent Repository

Permanent repository is where all the AIP are actually stored. The metadata information (METS) is kept separated from the multimedia files with a link within the METS to the physical file location.

Rosetta default repository is disk (NFS) and can be configured to use other repositories.

4.2.4 Operational

Operational is the module that allows Rosetta staff members to update objects in the permanent repository. It keeps a copy of all the AIP information in an ORACLE table, structured in a set of tables that enables Rosetta to manage the maintenance of the AIP easily.

4.2.5 Management

A set of reports provides Rosetta staff members with needed information regarding the data currently in Rosetta and an administration/configuration capabilities needed by staff member in order to configure the system for different materials/flows.

4.2.6 Publishing

Rosetta publishes the AIP information for external usage using OAI-PMH protocol.

4.2.7 Delivery

A set of 'out of the box' viewers are plugged in which enable users to view the IE content. Additional viewers may be plugged in using the viewer SDK.

4.2.8 Preservation

The preservation module manages all aspects related to preservation and includes:

- Format/Application/Risk Libraries – Knowledge base to register and manages all formats (based on PRONOM), applications (based on PRONOM) and risks
- Risk reports – Based on risks created in the risk library, Rosetta produces a report that lists all the files associated with that risk
- Preservation plan – Based on risk reports output, a preservation plan can be defined for each of the risks. Preservation plan will examine few alternatives, evaluate (manually & automatically) each of the alternatives and will provide the user with all the information needed in order to decide what will be the best alternative to be used.
- Preservation execution – Rosetta will manage the process of running a preservation action for the entire population relevant for that risk

4.3 Rosetta Components Compliant with OAIS

The following diagram illustrates the relation between the OAIS components and the different Rosetta modules:

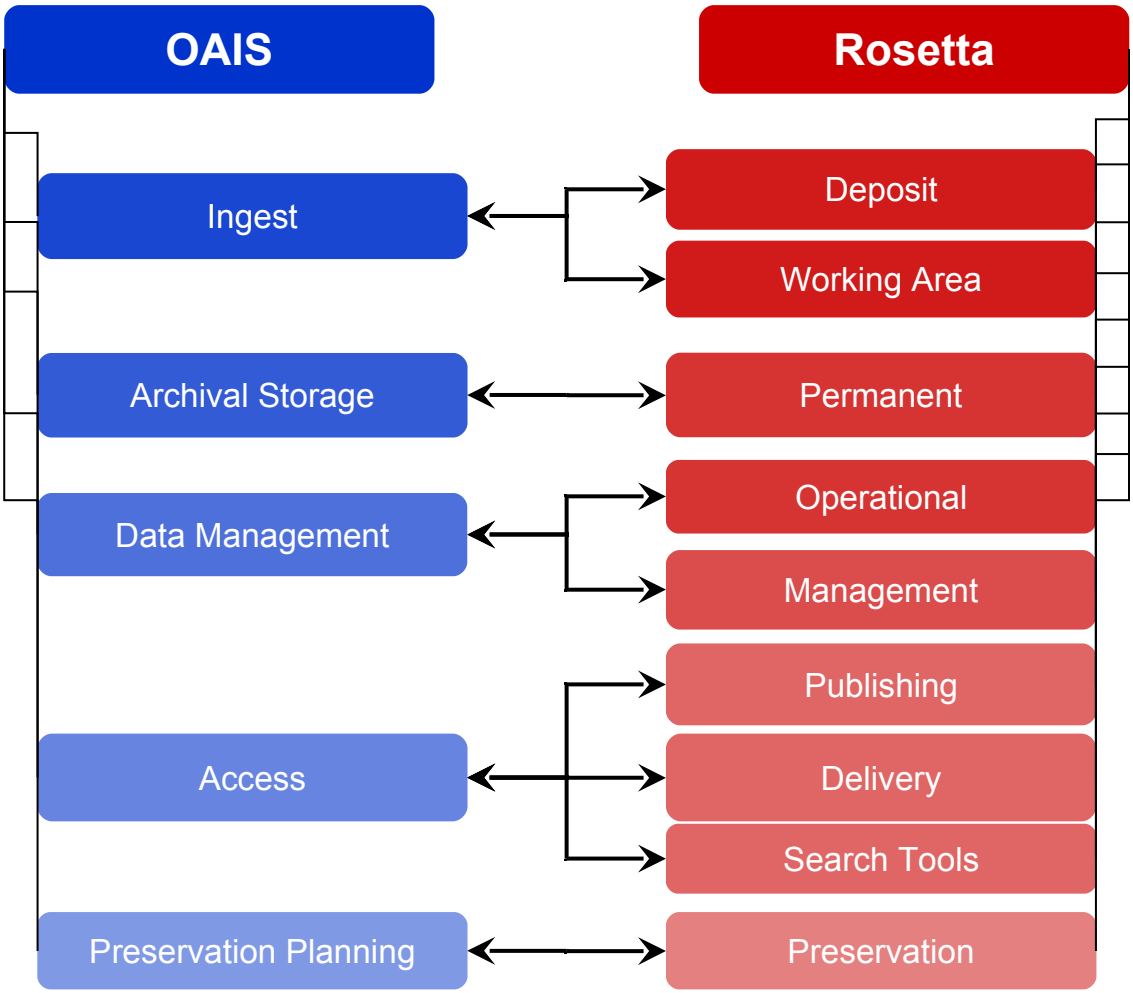


Figure 4: Rosetta components compliant with OAIS

4.4 OAIS Ingest Module

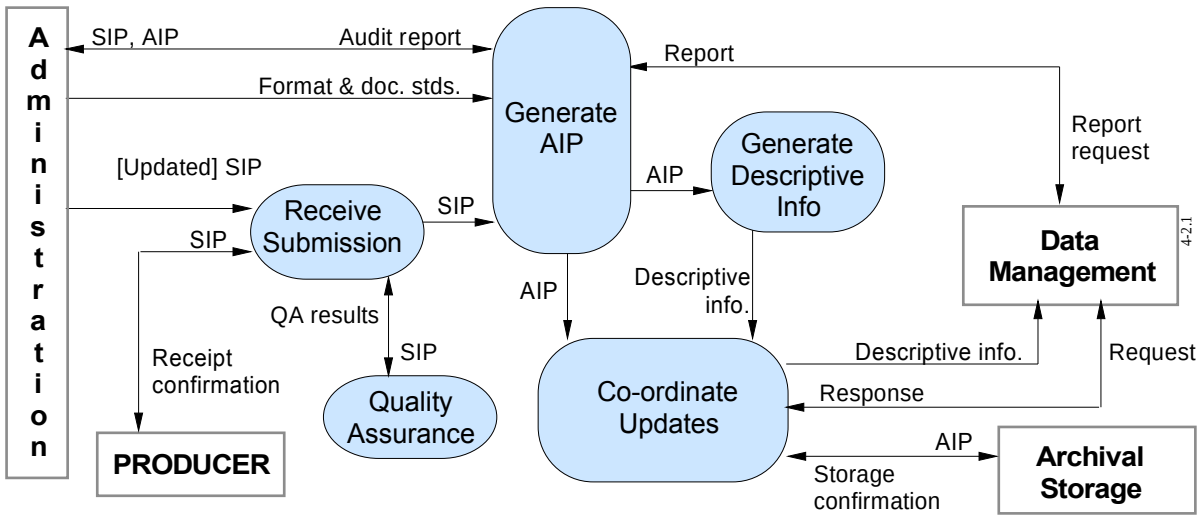


Figure 5: Functions of Ingest

The **Receive Submission** function provides the appropriate storage capability or devices to receive a *SIP* from the Producer (or from Administration). Digital SIPs may be delivered via electronic transfer (e.g., FTP), loaded from media submitted to the archive, or simply mounted (e.g., CD-ROM) on the archive file system for access. Non-digital SIPs would likely be delivered by conventional shipping procedures. The Receive Submission function may represent a legal transfer of custody for the Content Information in the SIP, and may require that special access controls be placed on the contents. This function provides a *confirmation of receipt* of a SIP to the Producer, which may include a request to *resubmit a SIP* in the case of errors resulting from the SIP submission.

The **Quality Assurance** function validates the successful transfer of the SIP to the staging area. For digital submissions, these mechanisms might include cyclic redundancy codes (CRCs) or checksums associated with each data file, or the use of system log files to record and identify any file transfer or media read/write errors.

The **Generate AIP** function transforms one or more SIPs into one or more AIPs that conform to the archive's *data formatting and documentation standards*. This may involve file format conversions, data representation conversions or reorganization of the content information in the SIPs. The Generate AIP function may issue *report requests* to Data Management to obtain *reports* of information needed by the Generate AIP function to produce the Descriptive Information that completes the AIP. This function sends *SIPs or AIPs for audit* to the Audit Submission function in Administration, and receives back an *audit report*.

The **Generate Descriptive Information** function extracts Descriptive Information from the AIPs and collects Descriptive Information from other sources to provide to Data Management. This includes metadata to support searching and retrieving AIPs (e.g., who, what, when, where, why) and could also include special browse products (thumbnails, images) to be used by Finding Aids.

The **Coordinate Updates** function is responsible for transferring the AIPs to Archival Storage and the Descriptive Information to Data Management. Transfer of the *AIP* includes a *storage request* and may represent an electronic, physical, or a virtual (i.e., data stays in place) transfer. After the transfer is completed and verified, Archival Storage returns a *storage confirmation* indicating (or verifying) the storage identification information for the AIP. The Coordinate Updates function also incorporates the storage identification information into the *Descriptive Information* for the AIP and transfers it to the Data Management entity along with a *database update request*. In return, Data Management provides a *database update response* indicating the status of the update. Data Management updates may take place without a corresponding Archival Storage transfer when the SIP contains Descriptive Information for an AIP already in Archival Storage.

4.4.1 Implementation in Rosetta – Deposit Module

The Deposit module is the primary interface for submitting material into the Repository. It is used both for external and for staff-mediated deposits.

Material is submitted through institution-defined **Material Flows**, which determine the deposit UI for a specific material type as well as the associated content structure and content handling parameters. The following deposit content structures will be supported in phase I:

- Set of files

- Directory Structure
- MARC21 (incl. pointers to files)
- METS
- CSV (Comma-Separated Values incl. pointers to files)
- Dublin Core (DC)
- MODS

Material Flows are associated with Producer Profiles. The Producer Agent inherits the Material Flows associated with the Producer Profile when depositing that Producer's material.

Material Flows can be generic or personalized. **Generic Material Flows** can be associated with any Producer type (Casual, Registered, Trusted, Internal), whereas **Personalized Material Flows** are Producer-specific and are reserved for Trusted or Internal Producers only..

The DPS Deposit module supports two basic types of Material Flows in phase I, Manual and Automated.

Manual Material Flow wizards consist of the following standard five steps:

1. Material Flow Description screen
2. Descriptive Metadata form

   Any number of Descriptive Metadata forms based on the Dublin Core format can be defined by the institution. Form fields can be defined as updatable, read-only, or hidden. Default values as well as controlled lists can be mapped to each field. Fields can be marked as mandatory or optional, and can furthermore be associated with validation procedures.

3. Access Rights / Assertion of Copyright form

   This form consists of an optional selection list of institution-predefined access rights definitions, from which the Producer Agent can select the appropriate option, as well as of a Boiler Plate statement.

4. Upload Files form

   Three different types of submission formats will be supported:

   - General: Files are uploaded one at a time based on a single file profile, which controls the file type (local or URL), the allowed file extensions/MIME types, and the maximum file size.
   - Detailed: Files are uploaded one at a time based on a list of file profiles.
   - Bulk: A set of files are selected and uploaded as one batch based on a single file profile.

5. Deposit Confirmation screen

Manual Material Flows can either be Generic or Personalized.

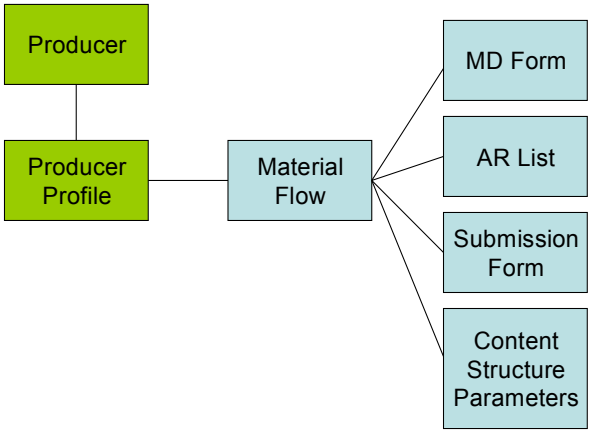


Figure 6: Manual Material Flow High Level ERD

Automated Material Flows are one-stop flows: The Producer Agent selects the appropriate Material Flow and confirms the submission to the deposit. Manual Material Flows require the Producer Agent to explicitly specify the files to be included in the deposit. Automated Material Flows are associated with a preset NFS or FTP location, from which the Staging server should retrieve the files as the first stage of SIP Processing.

(The original file path will be stored as part of the Administrative Metadata, in the DNX format. See the DNX FSD for further details)

Automated Material Flows are usually Personalized, i.e. suited to Trusted (preset FTP location) and Internal (preset NFS location) Producers. However, it is possible to define a Generic Automated Material Flow that can serve Staff Mediated deposits for multiple Producers.

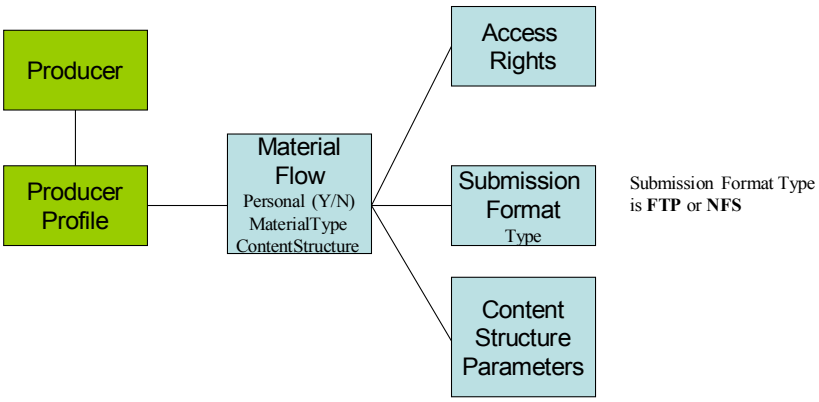


Figure 7: Automated Material Flow High Level ERD

A **Deposit Activity** record is generated in the Deposit Area as a result of the execution of a Material Flow. A SIP is derived from the Deposit Activity record as part of the SIP Submission. (For more details, see the SIP Submission FSD and the SIP Processing FSD)

The ID assigned to the SIP by the Staging Server is returned to the Deposit client, which registers it in the relevant Deposit Activity record for later reference.

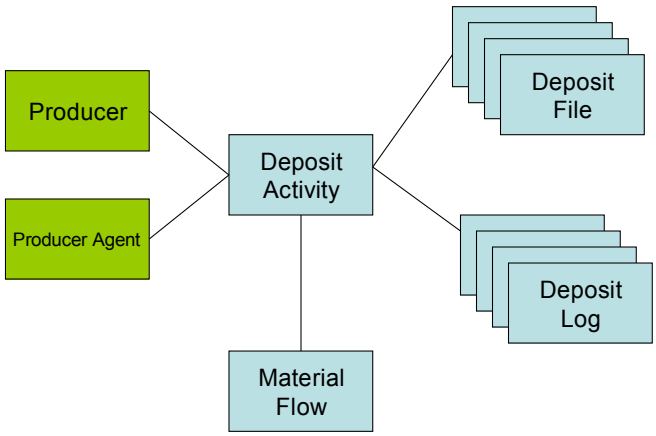


Figure 8: Deposit Activity High Level ERD

Deposit Activities are organized into folders according to their status:

- Draft Folder (deposits saved in the Deposit Area for later completion, pending submission)
- Submitted Folder (open deposits, pending approval)
- Approved Folder (closed deposits, approved)
- Declined Folder (closed deposits, declined)
- Returned Folder (open deposits, pending resubmission)
- Resubmitted Folder (open deposits, pending approval)

The Producer Agent can review the Deposit Activities and the associated log entries (status updates and communications between the Producer Agent and the approver staff) through the Deposit Module at any stage.

4.4.2 Material Flow Template, Content Structure and Content Acquisition Method

A Material Flow determines the user interface through which material can be submitted to the digital archive, as well as the expected content structure of the submitted material and the matching content acquisition method (HTTP uploading to the Deposit Area, or redirecting to FTP or NFS location). It moreover includes a set of technical attributes for the transformation of the Submission Content into the standard METS SIP format as expected by the Staging Server, as well as some information for the routing of the SIP to the appropriate Staging workflow/workspace.

Two Material Flow templates will be supported in phase I:

1. **Manual**
   Corresponds to the standard 5-step material deposit wizard (Step 1: Selection of Material Flow; Step 2 : Descriptive Form; Step 3: Access Rights/Assertion of Copyright; Step 4: Upload Files; Step 5: Deposit Confirmation)

2. **Automated**
   Automated Material Flows are associated with a preset FTP/NFS location from which the files are retrieved by the Deposit Application upon submission of the deposit. Steps 2 through 4 are skipped by the material deposit wizard. The Deposit Confirmation screen opens when the User selects an automated Material Flow.

A 3-dimensional matrix controls the various combinations of content structure and content acquisition methods supported for each template. This matrix is maintained by Ex Libris and is not accessible for customization. It reads as follows in phase I:

	Manual Template			**Automated Template**		
	HTTP Upload	FTP	NFS	HTTP Upload	FTP	NFS
Set of files	X					
CSV					X	X
Directory Structure					X	X
Dublin Core (incl. pointers to files)					X	X
MARC21 (incl. pointers to files)					X	X
METS					X	X
MODS (inc. pointers to files)					X	X

4.4.3 Material Flow Management Contexts and Associated Roles

Material Flows are associated with Producer Profiles. They can be generic or personalized. Generic Material Flows can be associated with any Producer Profile type (Casual, Registered, Trusted, Internal), whereas Personalized Material Flows are producer-specific and are reserved for Producers of type Trusted and Internal.

Material flow management can thus be invoked in three different contexts:

- **Global**

Management of the Institution's Material Flow pool
Role: Deposit Manager

- **Generic Producer Profile** (Casual and Registered Producers)

Management of the Material Flows associated with a generic Producer Profile
Role: Deposit Manager

- **Personalized Producer Profile** (Trusted and Internal Producers)
Management of the Material Flows associated with a specific Producer
Role: Negotiator

4.4.4 Material Flow Building Blocks

Material Flows are built by assembling following reusable building blocks, which are freely configurable by the Institution:

- **Metadata Forms** (Step 2 of the Manual Material Flow Wizard). Metadata forms are based on an expandable Dublin Core (DC) format. Their fields can be defined as input, read-only, or hidden. They can be assigned default values as well as form-specific or generic lookup lists in the case of controlled fields. Out-of-the box validation procedures (URL, ISBN, and ISSN) can be associated to each form field, which can also be flagged as mandatory or optional.

- **Lookup Lists** Step 2 of the Manual Material Flow Wizard). These lookup lists can be associated with any controlled metadata form field (field type defined as Combo, Radio-box or Check-box).

- **Access Right Options** (Step 3 of the Manual Material Flow Wizard)

- **Copyright Statement Boilerplates** (Step 3 of the Manual Material Flow Wizard)

- **HTTP Upload Submission Formats** (Step 4 of the Manual Material Flow Wizard)
Following submission format types are supported for HTTP uploading:
    - General (based on a single file profile: file location, allowed file extensions, minimum number of files, maximum number of files, maximum file size)
    - Bulk (based on a single file profile - see above)
    - Detailed (separate file profiles for each file upload option: file location, allowed file extensions, mandatory flag, maximum file size)

- **Content Structure Templates.** A different set of mapping/transformation parameters (such as Field Position → DC tag mapping for CSV formats, or tag of field containing the file stream location for DC, MARC21, and MODS) is associated with each content structure supported by the Deposit Application (see matrix above). A separate content structure template with the relevant parameter values must be set up for each variant content structure format as negotiated by the Library with its Producers.

These generic Material Flow building blocks are managed independently from any Material Flow and must be created ahead of time as the Material Flow editor doesn't allow for on-the-spot creation/updating of them.

Personalized Material Flow building blocks (Metadata forms and submission formats), in contrast, are intrinsically parts of the personalized Material Flow and can only be created and updated from the matching (personalized) Material Flow. When a personalized Material Flow is deleted, all associated personalized building blocks are deleted in the process.

When personalized Material Flows are derived from existing Material Flows (Generic or Personalized), the ID of the original Material Flow is kept in the SourceID field. (This field is not displayed in the UI.)

4.4.5 Deposit API

A set of Web services, Java code utilities, and command-line scripts will be provided for users who are interested in depositing materials directly into the Deposit Server without going through the provided Deposit UI/Client. This programming package is referred to as the DPS SDK (Software Development Kit).

In order to submit materials via the Deposit API, the user will first have to create and organize the material in the prescribed format and directory structure. In order to support such activities, the DPS SDK includes a set of java classes that allow the programmatic creation of METS based xml files.

Once the material is organized, it can be submitted via the Deposit API. Sets or batches of deposits can be submitted together, though each one is handled discretely as a separate deposit activity.

In order to activate the Deposit API, certain parameters are necessary for security and application processing of the material. This information is necessary for the processing of the deposited materials, the most important of which is the Id of the Material Flow through which the deposit should be conducted. Deposits using the Deposit-API must be done only via automated Material Flows. In order to allow programmatic construction of the call to the Deposit API, an additional set of "helper" APIs and java client classes is included in the DPS SDK.

When the Deposit API is invoked, the Deposit Server will validate the provided information and if no errors are discovered, it shall proceed to process the deposited materials as if they were deposited via the Deposit Client. In the event of errors, the deposited materials will be treated in an identical way to materials deposited via the Deposit UI. No special report will be sent back to the caller of the Deposit API. The resubmission of rejected SIPs or files will be done only via the regular Deposit UI.

In addition to the APIs discussed above, the DPS SDK will also expose an API for querying the processing status of a specific deposit activity. The user may use this functionality or check the processing status of a submitted deposit activity via the regular Deposit UI.

4.4.6 Validation Stack - Solution Overview

The main use of the Validation Stack is as part of the Pre-approval stage. All files that are successfully loaded to the Staging Server must pass several validation checks. These validation checks are bundled together in a process called the Validation Stack. The Validation Stack consists of:

- Several validation checks
- An activity for extracting the technical metadata from the files
- Activities for updating the system with the results of the process.

The Validation Stack is the first step in the staging server. Only files that successfully pass the Validation Stack checks can continue to the next steps (i.e. Assessor and Re-arranger or Approver) as configured in the SIP Processing configuration. Files that failed in one or more of the validation checks are marked as error and need to be handled by the Technical Analyst.

The Validation Stack can also be invoked by:

- A staff user, using the Web Editor or the Technical analyst workbench.
- Directly using the Process Automation module.
- As a scheduled job in the permanent repository.
- Whenever a file (stream) is replaced

The following activities are performed as part of the Validation Stack:

- Fixity Check - checks the integrity of files, to verify it has not become corrupted.
- Format Identification – determines the format to which a file conforms.
- Virus Check – checks whether the file is infected by a virus or not.
- Technical Metadata Extraction – extracts technical metadata from the file (such as: size, type, creation date etc.) to be stored in the system.
- Update Information – this is the last activity performed in the Validation Stack. This activity updates the relevant operational entities with the results of the Validation Stack process.

Not all the activities are necessary whenever the Validation Stack is invoked, for example: during scheduled run of the validation stack on the permanent repository there is no need to update any operational entities.

4.4.7 Quality Assurance – Human Stage - Solution Details

According to the SIP routing rules, each SIP is assigned to an Approval Group. The Approval group is assigned using the SIP Processing rules. The rule inputs include:

- Material Flow
- Material Type
- Producer Type
- Producer Group
- Producer

Each role of Assessor, Arranger, and Approver is part of an Approval Group. When one member logs in to the Workbench, they can see the SIPs that are assigned to their Approval Group.

Note: A user having all three roles (i.e. Assessor, Arranger, and Approver) will see the relevant SIPs for each stage when entering each Workbench. For example, when a user with both Assessor and Arranger roles enters the Assessor Workbench, he or she sees all the SIPs that are currently assigned to the Assessor. If he or she enters

the Arranger Workbench, only SIPs assigned to the Arranger stage and belonging to that Approval Group are present.

The Approval Group value can be overridden only by a privileged user (which has the privilege “Assessor/Arranger/Approver - Full”). The Typical Assessor/Arranger/Approver cannot forward or reassign the SIP to a different group.

All material submitted by Registered Producers that are not Trusted is reviewed manually by Assessors and Re-arrangers to ensure that the SIP’s content is appropriate and suitable for preservation.

Deposits from Trusted and Internal Producers are sampled-based on a pre-defined Sampling Rate.

4.4.7.1 Sampling Rate Determination

The following rules determine whether the SIP will go to the Approver:

1. If there is a specified Sampling Rate for this Material Flow, then all SIPs that were deposited through this Material Flow will be sampled by this rate.
2. If there is no specified Sampling Rate for the Material Flow but there is a Sampling Rate for this Producer Profile, then the SIPs that are deposited through this Producer Profile will be sampled by this rate.
3. If the SIP is deposited and neither the Material Flow nor the Producer Profile has a Sampling Rate value, the SIPs will be sampled at a rate that is pre-defined.
4. If at least one of the SIP files was ignored or declined the sampling rate will be 100%, therefore, all such SIPs will go thorough the Approver Workbench.

The actual sampling is done by calling a method for each SIP that returns a random number according to the given rate. For example, for a certain Processing Configuration, when the Sampling Rate is set to 20%, the system randomly chooses a number between 1 and 5 and all SIPs with number 1 will be sampled by the Approver.

4.4.8 Workbench Details

For each approval stage, a list of SIPs that is currently waiting for processing is displayed. The user can sort the list by clicking the heading of one of the required columns. The user can search the SIPs by Producer, Title, Material Type and PID. The user can filter the SIPs list by the following options:

- Assigned to me
- Not assigned
- Material Type

For each SIP, the user can

- review the SIP information and decide if the SIP or any of its IEs should be approved, rejected, or declined.
- add an internal note that is logged on the SIP level.

The SIP information is presented on four tabs:

1. SIP Content – list of IEs. The tab contains the list of IEs with the relevant actions that can be performed by the user.
2. SIP History- this tab contains the SIP history including all the events that has been logged during the SIP processing.
3. SIP Notes – this tab contains the SIP Internal Note and alerts (if exists). The Internal Note is a free text log that each user can view and add his own text, the alerts are automatically generated by the system in the following cases:
    - The Technical Analyst declined one or more files from this SIP.
    - The Technical Analyst ignored technical problems.
4. SIP Metadata – this tab displays the SIP metadata in read-only mode.

At each tab, the user can decide what should be done with the SIP and/or its IEs. The following actions are available for the user according to his role:

SIP Level Actions:

- Move To Next Stage – The SIP will be moved to the next stage according to its SIP processing configuration. For example, if the SIP is currently in the Approver stage, the SIP will move the next stages which means – Enrichment and Move to Permanent.
- Reject SIP – The SIP Status is set to 'Rejected' and so is the Deposit Activity Status. The Producer Agent will get the SIP back for processing. The user will be able to choose the Reason and add Notes for his action.
- Decline SIP – The SIP Status is set to 'Declined' and so is the Deposit Activity Status. The user cannot do anything with this Deposit Activity (It cannot be re-submitted). The user will be able to choose the Reason and add Notes for his action.
- Move to Assessor/Arranger/Approver pool – the user can move the SIP to a different approval group for further handling.
- Move to a specific Assessor/Arranger/Approver - the user can move the SIP to a specific user in a different approval group for further handling.
- Assign To – the user can assign a SIP to a different user for doing so - the following rules should apply:
    - If the SIP is already assigned to a user, only a privileged user (Assessor/Arranger/Approver - full) can reassign it.
    - If the SIP is not assigned to anyone, the user (typical or full) can assign the SIP to any other user.
    - If the SIP is assigned to a specific user, the same user can unassign it and then assign it to someone else.
- Forward – the user can forward the SIP to a different user (the same as Assign To but does not require special privilege and can be done only from SIP View screens)

IE Level Actions:

- Reject IE – The Producer Agent will get the SIP back for processing and will have only the ability to re-submit the rejected IE. The user will be able to choose the Reason and add Notes for his action.

- Decline IE – The user cannot do anything with this IE (It cannot be re-submitted), but the SIP can continue to the next stages. The user will be able to choose the Reason and add Notes for his action.

- Assign CMS ID – The user can link between an external CMD ID and a specific IE or group of IEs in the DPS.

- Assign Access Rights Policy – The user can click this button to assign access rights policy.

The action Assign CMS ID is available as group action (marking checkboxes and using the action combo box) only to the Arranger.

4.4.9 SIP Submission - Solution Overview

Users submit materials to the DPS via the Deposit UI (a.k.a. Deposit Client). As soon as the user requests to add a deposit activity, the Deposit UI contacts the Staging Server and requests that a new SIP Id will be generated. The Staging Server relays the request to the SIPIdGenerator component, which is responsible for issuing a SIP Id and registering the new Id in the SIP Registry. The generated SIP Id is returned to the requesting Deposit UI.

The Deposit UI generates the screens for the deposit process based on the information found in the Deposit Data Model and the user's selections. Once the user decides to submit the materials to the system, the Deposit UI submits the deposit activity by writing an entry in the table HDpDepositActivity.

On the Deposit Server, a scheduler called Quartz is activating a thread called Deposit Worker every 5 seconds. This thread checks the table if there is any Deposit Activity that is waiting to be processed.

The Deposit Worker starts the Submission process for each Deposit Activity. First, the Deposit Worker generates a Load Directory for the submitted deposit activity. A single Load Directory is created per deposit activity on a shared NFS storage. The Load Directory will hold the row deposited materials as well as all of the information that will be generated by the Deposit Worker for the deposited activity throughout its processing.

The Deposit Worker then proceeds to acquiring the submitted content. The submitted content is acquired either a-synchronically, from a remote FTP directory (as will be the case with large publishers), or synchronically, from the depositor's local disk (as will be the case with casual depositors). In both cases the system performs fixity check to the files (MD5) as part of the acquiring and stores the value for future comparisons. The acquired submission content is written to the relevant Load Directory and stored in the Content sub-directory. For remotely acquired content (e.g. FTP or NFS), it should be noted that the content is either moved or copied based on the handling method parameter defined in the Submission format associated with the Material flow used.

The streams are stored hierarchically under the Load_{Deposit_Activity_Id}>Deposit>Content>Stream subdirectory. If the acquisition of the deposited materials fails (completely or partially), the system will either reject

the deposit activity (in which case, the rejection will be reflected in the Deposit UI) or will route the problematic SIP to the Technical Analyst's inbox, depending on the defined rules (henceforth: error handling mechanism).

Once the acquisition of the files is successfully completed, the Deposit Worker continues to the Validation before the Translation stage. In the Translation stage, the submitted content, which must be in one of the formats that are supported by the system, is converted into one or more system formatted METS files. In order to make sure the Translation phase will work on valid files, the Pre-Translation Validation will check that the streams are located where they should be according to their pointing (dc:identifier, tag 856 in MARC files) and that the encoding of the content is as it should be (e.g. UTF-8).

In case the content structure fails this Validation, the SIP Submission of this SIP will stop and the Deposit Activity will be returned to the Producer Agent.

After the SIP passes this Validation successfully, the SIP will continue to the Translation stage. The conversion is done by a specializing converter (one converter per each supported submission format) and is guided by the Content Structure that was selected by the depositor or by the negotiator who constructed the material flow. Each generated METS file represents a single PREMIS Intellectual Entity(IE) and is implementing the PREMIS standard (i.e., the structure of the METS complies with the PREMIS standard). Since the submitted content may contain information about multiple IEs within a single file, a single submission content may yield multiple system METS. The generated METS are stored under the SIP>IES directory of the relevant Load Directory and contain pointers to the relevant files in the Stream sub-directory. A Translation failure will trigger the system's error handling mechanism.

After the successful completion of the Translation stage, the Deposit Worker will perform the Translation Validation stage. The purpose of the Translation Validation stage is to determine whether the Staging Server will be able to successfully process the generated METS. During the detailed design stage it shall be decided whether the Staging Server will provide the service for validating the generated METS (in which case, another API will be added to the Staging Server). A Validation failure will trigger the system's error handling mechanism.

Following the successful completion of the Validation stage, the Deposit Worker will proceed to the SIP Wrapping stage, during which the relevant Load Directory will be prepared for submission to the Staging Server. Thus, for example, the METS files within the SIP>IES directory are organized in zipped bulks to improve performance. A Wrapping failure will trigger the system's error handling mechanism.

When the system is done wrapping the SIP, the Deposit Worker will submit the SIP to the Staging Server using the Submit API. The Staging Server will then start processing the submitted SIP according to the business logic described in the SIP Processing spec.

Throughout the processing of the SIP in the Deposit Server, the system will store information about the deposited materials in the Deposit>Setting subdirectory. Provenance information, in contrast, will be stored directly in the relevant METS file (if the information is unique for the IE) or in the SIP>Shared subdirectory (if the information applies to all the IEs in the SIP).

Invalid SIPs that will end up in the Technical Analyst's (TA) inbox will be viewed and handled by the TA using the workbench that will be described in the TA spec. The TA

will be able to reject or decline the problematic SIPs (completely or partially) or to manipulate them outside the system (by downloading them) and resubmit them. Resubmitted SIPs will restart processing from scratch (i.e., from the point of entering the Deposit Server) and not from the point in which the error occurred/was discovered.

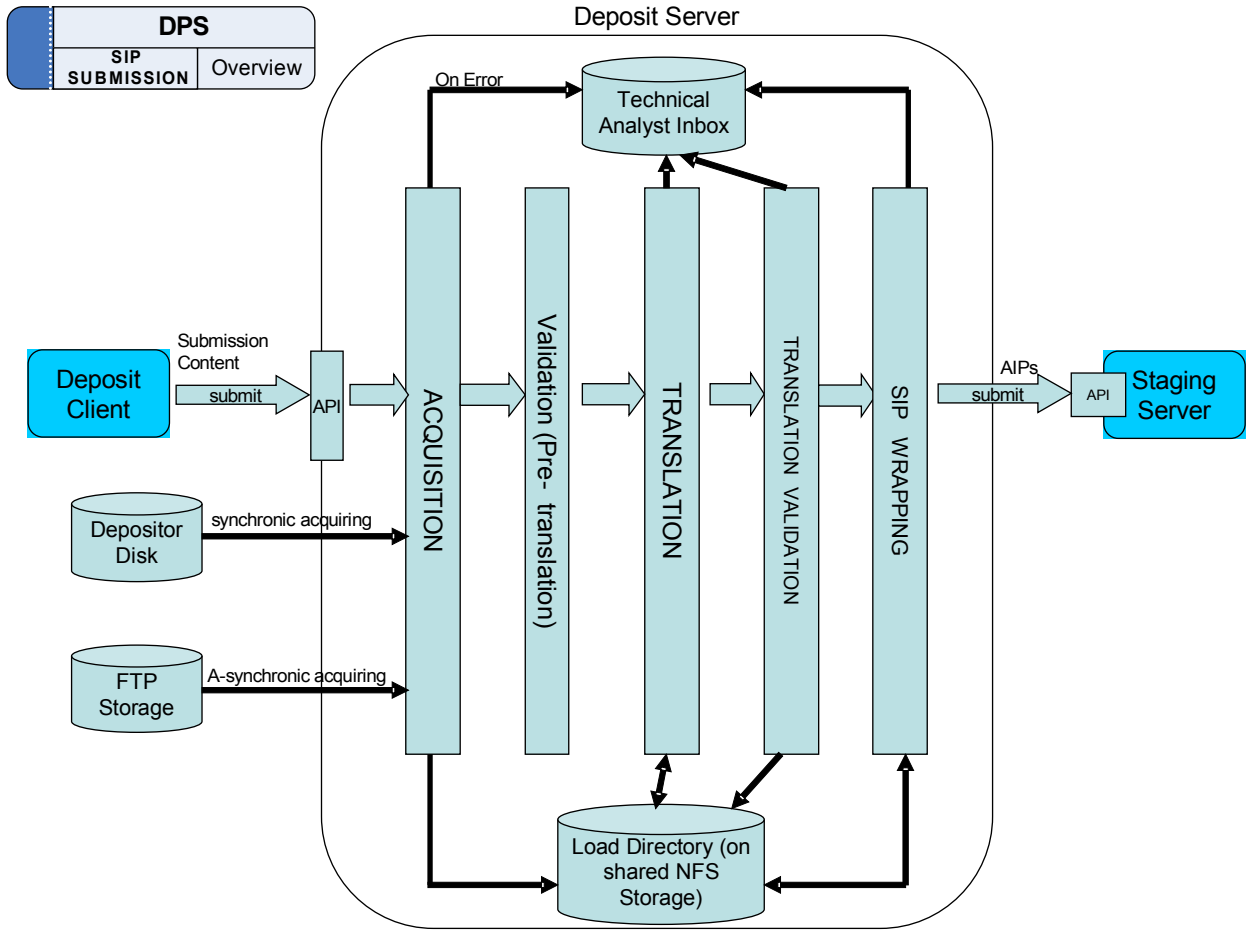


Figure 9: Technical Analyst Inbox

4.4.10 Handlers - Solution Overview

As described in the SIP Submission Specifications, a single Load Directory is created per deposit activity on a shared NFS storage. The Load Directory will hold the raw deposited materials as well as all of the information that will be generated by the Deposit Server for the deposited activity throughout its processing.

The Deposit Server then proceeds to acquiring the submitted content. The submitted content is acquired either a-synchronically, from a remote FTP directory (as will be the case with large publishers), or synchronically, from the Producer Agent's local disk (as will be the case with casual Producers). The acquired submission content is written to the relevant Load Directory and stored in the Content sub-directory. The streams are stored hierarchically under the Load_{Deposit_Activity_Id} > Deposit > Content > Stream > IE <n> subdirectory.

Once the acquisition of the files is successfully completed, the Deposit Server continues to the Translation stage. In the Translation stage, the submitted content, that must

be in one of the formats that are supported by the system, is converted into one or more system formatted DPS METS files. The conversion is done by a specializing converter (one converter per each supported submission format) and is guided by the Content Structure that was selected by the Producer Agent or by the negotiator who constructed the material flow. Each generated DPS METS file represents a single PREMIS Intellectual Entity(IE) and is implementing the PREMIS standard (i.e., the structure of the DPS METS complies with the PREMIS standard). Since the submitted content may contain information about multiple IEs within a single file, a single submission content may yield multiple DPS METS. The generated DPS METS are stored under the SIP>IES directory of the relevant Load Directory and contain pointers to the relevant files in the Stream subdirectory. A Translation failure will trigger the system's error handling mechanism.

4.4.11 Staging – Permanent Solution Overview

When the processing of a SIP in the Staging Server will be completed, the SIP's manifestation and the various Intellectual Entities (and all their subcomponents) contained within the SIP will be written to the Permanent Repository. Before an Intellectual Entity will be physically moved to the Permanent Repository, it may undergo (depending on the defined SIP Processing Configuration) further processing that is designed to insure that it contains all the information it should. Thus, for example, the IE may be enriched (e.g., by adding generated Thumbnails to it) and synced with external sources of information (e.g., with CMS). The move to the Permanent Repository constitutes the last stage in the SIP's Processing Configuration and marks the end of the SIP's life as an active element.

The Permanent Repository is responsible for the long term preservation of information that has been admitted to the system. To achieve this goal, the repository acts as a one way, "write once", digital "safe box", preventing the deletion, manipulation, and withdrawal of the information deposited to it. Users can therefore search the repository and view its various PREMIS objects but they cannot alter the content of the objects, nor can they withdraw them from this repository. An exception to the above rule is the limited ability of privileged users, under rare circumstances, to alter the content of stored objects (without leading to versioning) or to *physically* erase such objects. The allocation of such privileges will be done via the user management mechanism. Although not allowing the manipulation of stored information, the Permanent Repository will allow the admission of new versions to stored objects and will maintain all versions.

Periodic and occasional Maintenance services/jobs assure the readability and accessibility of the information contained within the Permanent Repository.

Since the reliability of the Permanent Repository is its most important characteristic, the Permanent Repository will be based on proven and mature technology, i.e., file system. The file system will be wrapped by a direct access layer and a services layer, creating a self-contained file-based dark (i.e., write once) repository. However, in order to enhance search and delivery performance, a database representation of the Permanent Repository's latest version of stored objects will be maintained (as a view over the Staging Server's repository).

4.4.12 Staging-Permanent Architecture

The Staging Server and the Permanent applications will be physically separated. Each application will manage its own storage (in a shared NFS), but will grant the other application reading privileges to that storage. Both applications will use the Repository database, although each will access the database via a separate view. The following diagram presents an overview of the Staging-Permanent architecture:

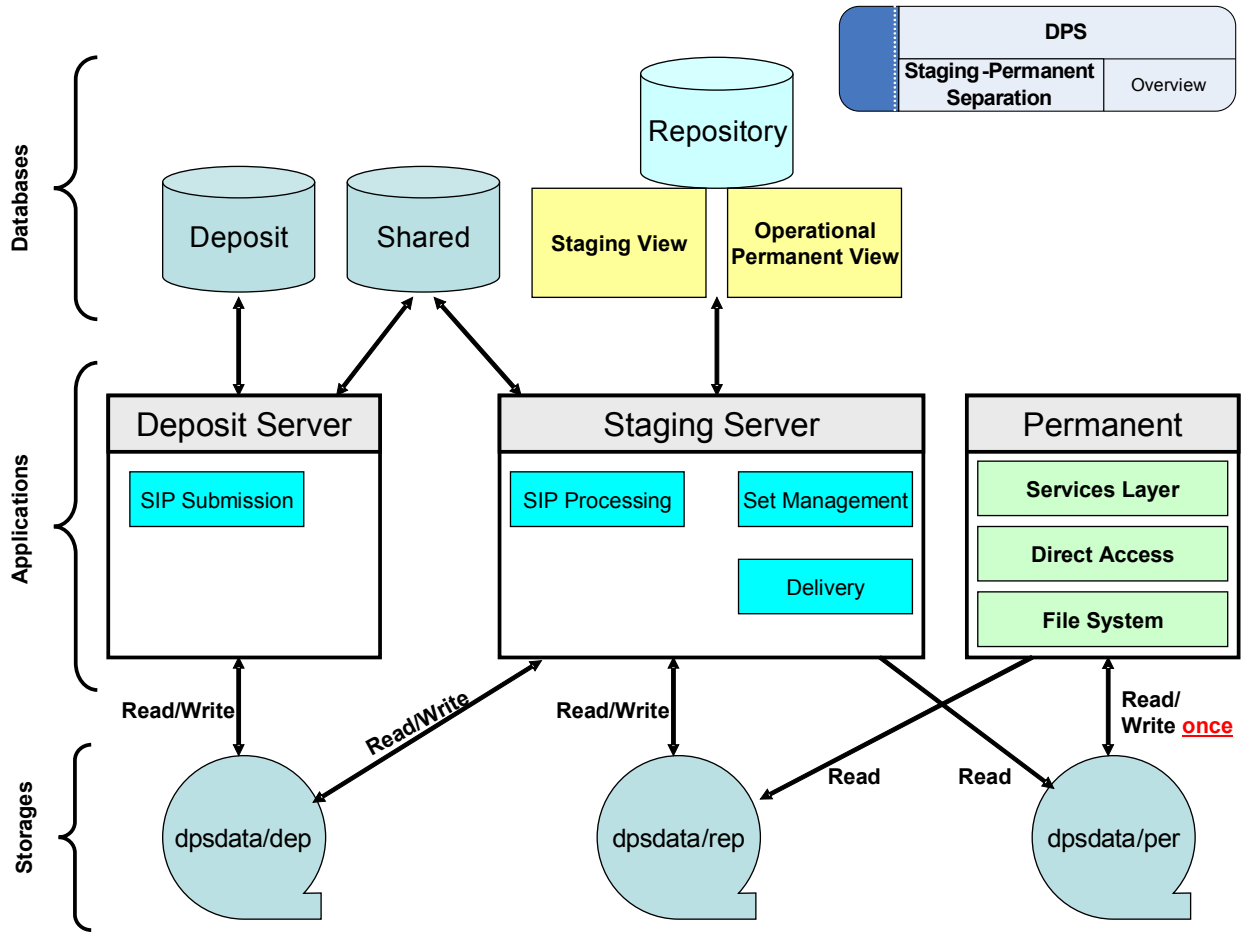


Figure 10: Rosetta Staging-Permanent Architecture

4.4.13 Permanent Repository Components

The heart of the Permanent Repository is the Permanent Application and the Permanent file-system based Storage. Together they create a self-contained and reliable repository for stored IEs and SIPs. These core components are complimented by the Operational Permanent View that offers a supportive layer for search and delivery functionality.

4.4.13.1 Permanent Application

The Permanent application is composed of three layers (top → down): Services layer, Direct Access layer, and File System layer. In phase I, the two top layers will of-

fer limited services, limited to the admission of objects into the Permanent Repository. The main responsibility of the Permanent application is to write IEs and SIPs that have been approved (in the Staging Server) for storage into the Permanent Repository's file-system based storage.

The application will be invoked by the Staging Server. When invoked, the application will write the information found in a designated directory in the Staging Server's storage into the Permanent Repository and will return a confirmation.

The Staging Server is responsible for placing the required information (i.e., the relevant content of SIPs that have been approved for storage) in the designated directory, for cleaning the directory once the storage in the Permanent Repository has been confirmed, for updating the Repository records to reflect the fact that the objects have been moved to the Permanent Repository and to purge from the Repository redundant information.

4.4.13.2 Permanent Application's API

The Services layer of the Permanent application exposes, in phase I, two APIs to support the admission of information into the Permanent Repository:

Service Name	Parameters (All of type String)	Returns
StoreIE	String ieXml – The IE XML	**Storage request confirmation**
StoreSIP	(String sipTarPath, Map sipAttributes) – The zipped files of the SIP_REGISTRY and the SIP_ITEMS records.	**Storage request confirmation**
storeSharedMd	(String mdXML, Map mdAttributes) - The shared MD that is stored separately of the SIP – Access Rights, CMS MD that is connected to multiple IEs.	**Storage request confirmation**
GetObject	**ID: The ID of the Intellectual Entity or Stream (i.e. PID) or SIP (i.e. SIPID)**	**Location information (e.g. Connection String, Full path to storage location)**

4.4.13.3 Storage

The Permanent Storage will be managed on a shared NFS to which the Staging Server application will also have access (for reading and writing purposes only, not deleting or updating). The storage will hold the streams and metadata of the stored IEs (and all their subcomponents) and the metadata of the stored SIPs. The storage will manage the deposited information according to the configurable storage rules.

The determination on where a given entity is stored is based on a set of storage rules. When an Intellectual Entity is stored in the Permanent Repository, it and its component streams are evaluated using the storage rules, and based on the outcome, allocated to a specific storage area (storage group).

For example, the customer would like to store all the video files (streams) on a dedicated server that have more storage space and specific backup and security policy,

different than the METS files that will be stored on smaller storage server which have more lightweight backup and security policy.

The evaluation is done based on various attributes of the Intellectual Entity, or in cases of streams on the full context of the stream within the File, Representation, and Intellectual Entity hierarchy.

4.4.13.4 Operational Permanent View

The Operational Permanent View is a view of the Repository, capturing only those records with Control.life_cycle attribute that equals 'Permanent'.

When a SIP is loaded into the Staging Server, control entries are created in the Repository to represent the PREMIS entities (i.e., IE, REP, FILE) that are contained within the SIP. The life_cycle attribute of these records equals, upon creation, to 'Staging', representing the fact that the entities are processed by the Staging Server.

When a SIP is stored in the Permanent Repository (by invoking the Store API of the Permanent application), the life_cycle attribute of the relevant control entries is set, by the Staging Server, to 'Transferred' and later, once the storage has been confirmed, to 'Permanent'. Pointers to physical files (metadata and streams) that are associated with the control entries are also updated to reflect the new location of the files (as returned by the Store API). The control information about stored objects is not taken off the Repository to allow a more efficient search (via the Set Management UI) and delivery (via the Viewer) functionality.

The Operational Permanent View captures only objects that have completed their transfer to the Permanent Repository. Such objects are not reflected in the Staging View, which is the view through which the Staging Server accesses the Repository.

4.5 OAIS Archival Storage Module

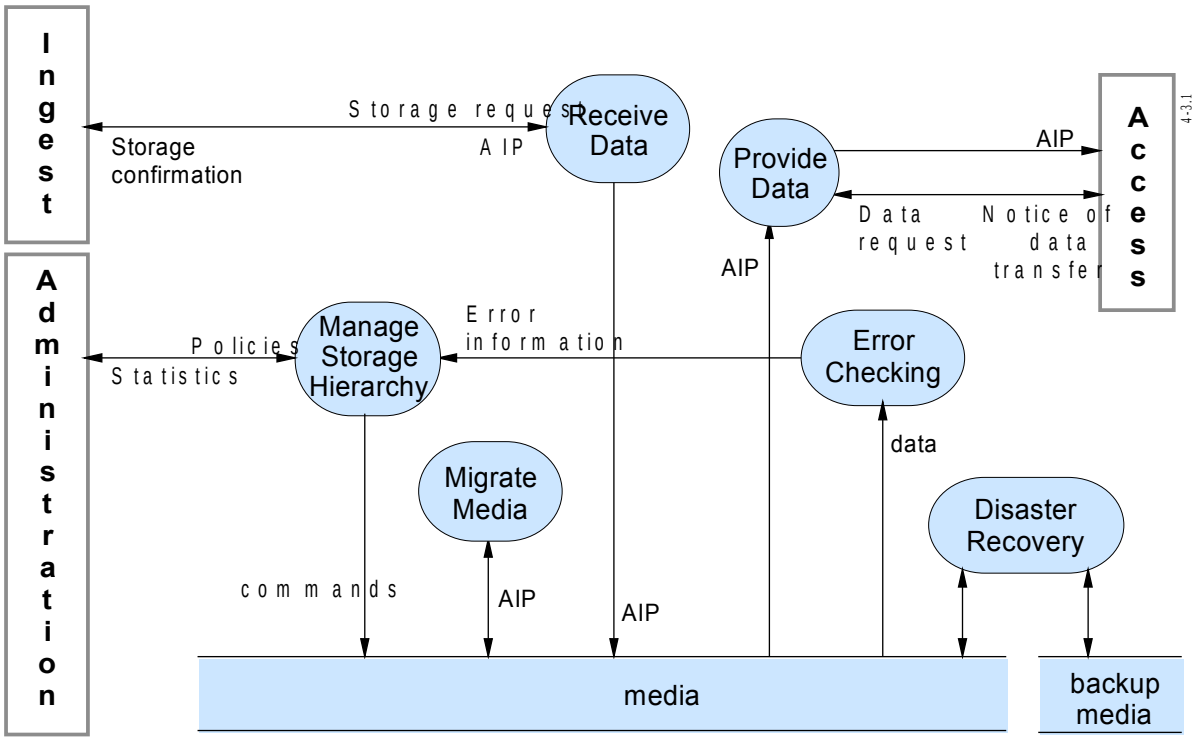


Figure 11: Functions of Archival Storage

4.5.1 Receive Data

4.5.1.1 Receive Data Requirements

The **Receive Data** function receives a *storage request* and an *AIP* from Ingest and moves the *AIP* to permanent storage within the archive. The transfer request may need to indicate the anticipated frequency of utilization of the data objects comprising the AIP in order to allow the appropriate storage devices or media to be selected for storing the AIP. This function will select the media type, prepare the devices or volumes, and perform the physical transfer to the Archival Storage volumes. Upon completion of the transfer, this function sends a *storage confirmation* message to Ingest, including the storage identification of the AIPs.

4.5.1.2 Implementation in Rosetta

The transfer of objects from the Staging Server to the Permanent Repository is a multi-step process, controlled by the Staging Server and executed via Task Chains (see Task Chain term). The steps that compose this process are:

Enrichment

During the enrichment step, the IEs that are to be stored are enriched. Thus, for example, the system may create Access Copies and Thumbnails for IEs that require such additions. Enrichment actions are performed via the execution of designated Task Chains. SIPs that have failed enrichment (i.e., SIPs that contain one or more IEs that have failed enrichment) will be routed to the TA as a whole (the IEs that have passed enrichment will not be separated from the failed IEs and will not continue to the next phase).

DPS-CMS Sync

After the successful completion of the enrichment step, IEs that are to be stored and have been assigned (by the Arranger/Assessor) a CMS id (see in Consolidated Approval Workbench spec), will be synchronized with CMS. The relevant parts of the IE's metadata will be overridden with the imported CMS metadata. The IE's original metadata (i.e., the metadata with which it has been deposited to the system) will still be preserved at the SIP's level (as the SIP that will be stored in the Permanent Repository will contain the original Deposit Content).

Data Transfer

After the successful completion of the DPS-CMS sync step, the information to be stored will be written to the Permanent Repository. For stored **IE**s, the information that will be passed to the Permanent application consists of all of the IE's (and contained sub-components) streams and metadata files. For stored **SIP**s, the information that will be passed to the Permanent application consists of the SIP's metadata, including the original content that was deposited to the DPS (as stored in the Load_{Deposit Activity Id}\Deposit\Content sub-directory).

The Data Transfer step will begin with the Staging Server invoking the Permanent application StoreIE API. Once invoked, the Permanent application will create an entry in a tracking table that will hold the information of the IE. Then it will send a

confirmation message to the Staging server confirming that the IE is in process of being copied. After sending the message, Permanent application accesses the supplied path and copies the information. When IEs and their component streams are transferred, they will be stored in the correct storage location in the Permanent Repository according to the configured storage rules. The data integrity and safety of the deposited objects will be checked after storing, by executing the Validation Stack on them. If the deposited objects will successfully pass the Validation Stack, they will be “committed” to the Permanent storage.

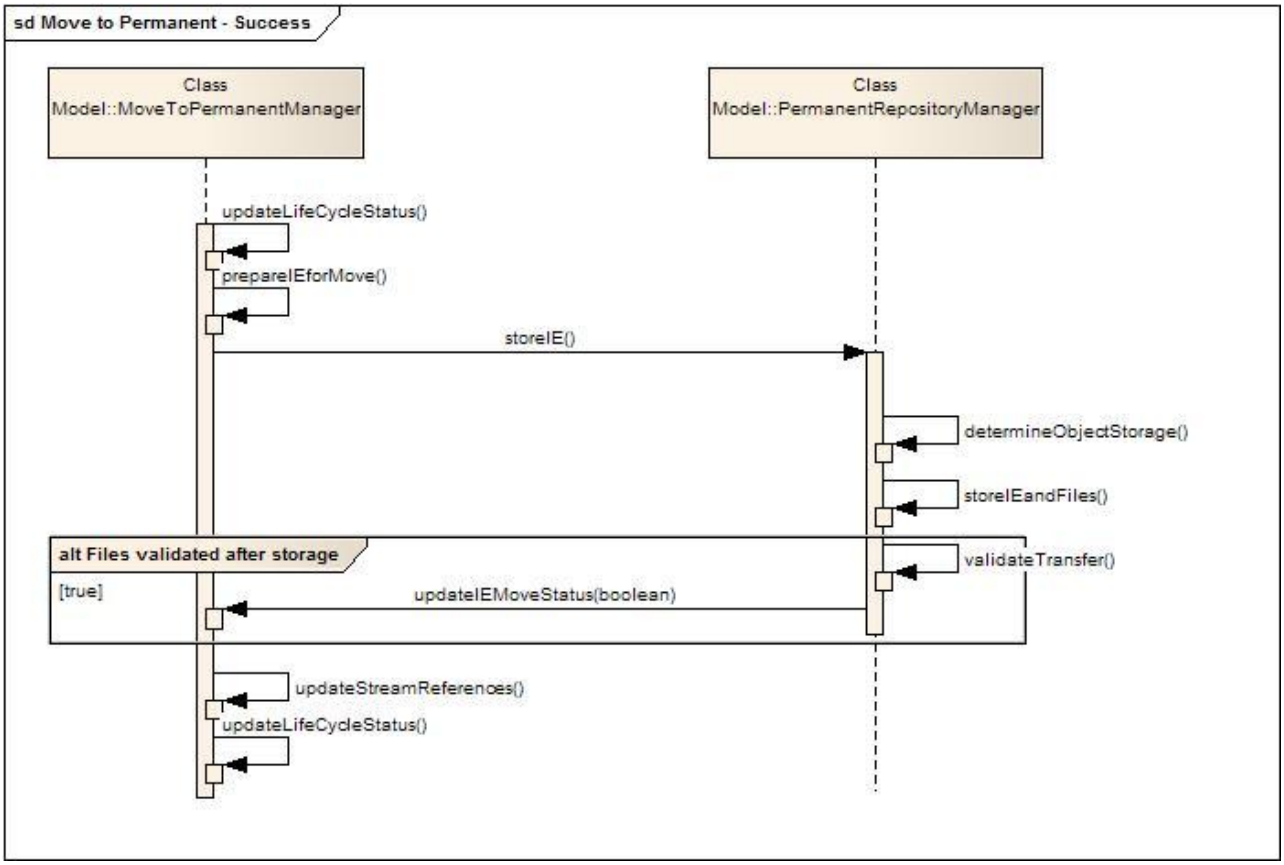
In case the copy process was interrupted, the tracking table will be used for recovery – The Permanent application will check which entries are still in the table and re-initiate the copy process of these IEs.

After copying the IE, the Permanent application will copy the SIP (in a zipped file) and the shared Metadata, both stored separately based on their Storage rules.

Only after successful copy of IE (and its sub-components) the tracking table entry will be deleted and the Permanent application will return a storage confirmation to the invoking Staging Server. Upon receiving the confirmation, the Staging Server will access the Repository (via the Staging view) and will change the life_cycle attribute of the transferred objects to 'Permanent' as well as purge the streams of the transferred objects from the Repository.

Note: The Permanent Repository API GetStorageLocation will be used by the Operational Permanent View in order to translate a File’s Permanent storage location in to an actual path for purposes of retrieval and delivery.

Move to Permanent – Successful



Move to Permanent – Failure

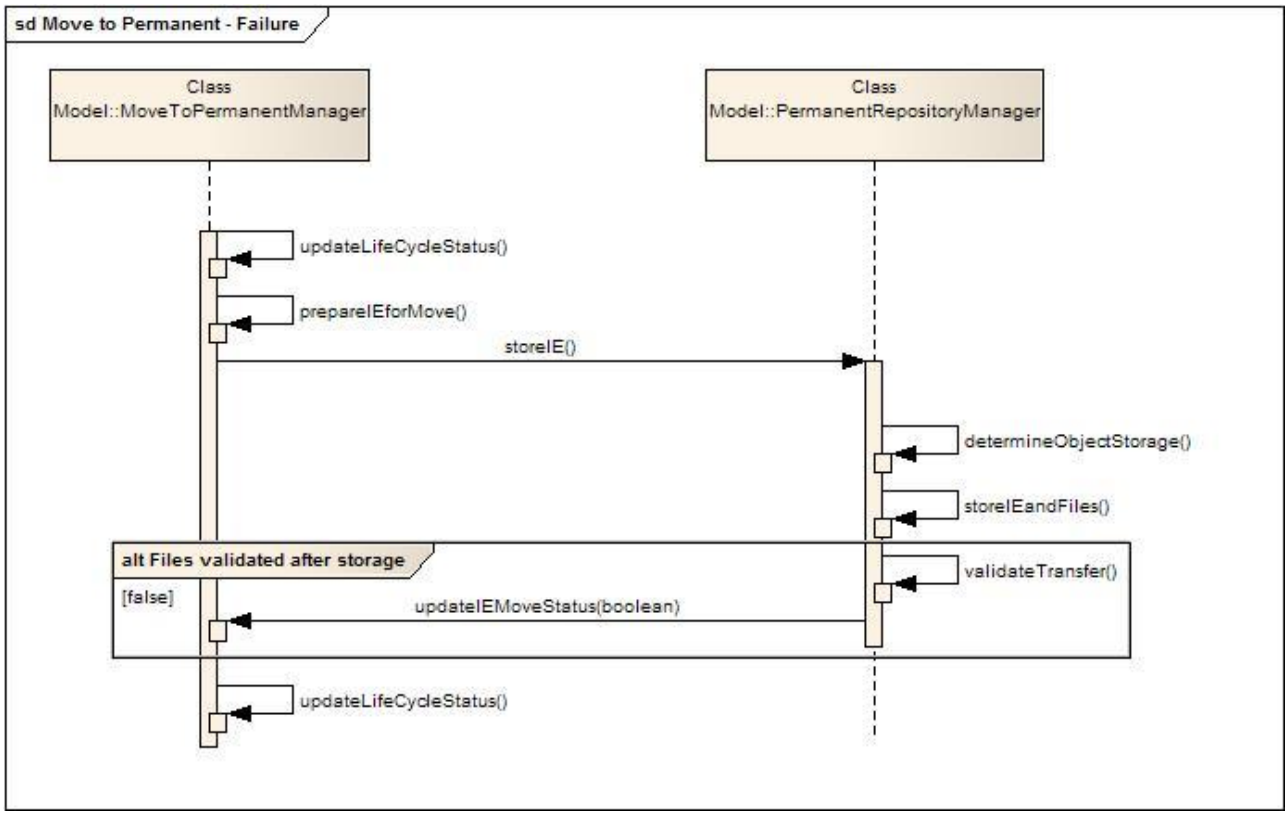


Figure 12: 'Move To Permanent' Flow

4.5.2 Manage Storage Hierarchy

4.5.2.1 Manage Storage Hierarchy Requirements

The Manage Storage Hierarchy function, positions via commands, the contents of the AIPs on the appropriate media, basing on storage management policies, operational statistics, or directions received from Ingest via the storage request. It will also conform to any special levels of service required for the AIP, or any special security measures that are required, and ensures the appropriate level of protection for the AIP. These include on-line, off-line or near-line storage, required throughput rate, maximum allowed bit error rate, or special handling or backup procedures. It monitors error logs to ensure AIPs are not corrupted during transfers. This function also provides operational statistics to Administration summarizing the inventory of media on-hand, available storage capacity in the various tiers of the storage hierarchy, and usage statistics.

4.5.2.2 Implementation in Rosetta

See 4.4.11

The Staging Server and the Permanent applications will be physically separated. Each application will manage its own storage (in a shared NFS), but will grant the other application reading privileges to that storage. Both applications will use the Repository database, although each will access the database via a separate view. The following diagram presents an overview of the Staging-Permanent architecture:

4.5.3 Replace Media

4.5.3.1 Replace Media Requirements

The Replace Media function guarantees to keep the capability to reproduce (i.e. to play media and take advantage of metadata) the AIPs over time. Within the Replace Media function the Content Information and Preservation Description Information (PDI) must not be altered. However, the data constituting the Packaging Information may be changed as long as it continues to perform the same function and there is a straight forward implementation that does not cause information loss. The migration strategy must select a storage medium, taking into consideration the expected and actual rates of errors encountered in various media types, their performance, and their costs of ownership. If media-dependent attributes (e.g., tape block sizes, CD-ROM volume information) have been included as part of the Content Information, a way must be found to reserve this information when migrating to higher capacity media with different storage architectures. Anticipating the terminology of section 0, this function may perform 'Refreshment', 'Replication', and 'Repackaging' that is straightforward. An example of such 'Repackaging' is migration to new media under a new operating system and file system, where the Content Information and PDI are independent of the file systems. However, complex 'Repackaging' and all 'Transformation' are performed under Administration supervision by the Archival Information Update function to ensure information preservation.

4.5.3.2 Implementation in Rosetta

The Meditor is used for editing IEs metadata, adding new Representations and running services such as "Thumbnail creation" on objects stored in the permanent repository. The Meditor plays a significant role in the migration process by acting as the tool used for adding the product of the migration process to the source IE. Meditor functions are invoked from the Workbench interfaces by staff users that want to

- Edit object attributes (via the DNX editor)
- Add or edit metadata records
- Perform object services such as: export and run validation stack
- Add representations
- Delete representations
- Upload or download files

4.6 OAIS Data Management Module

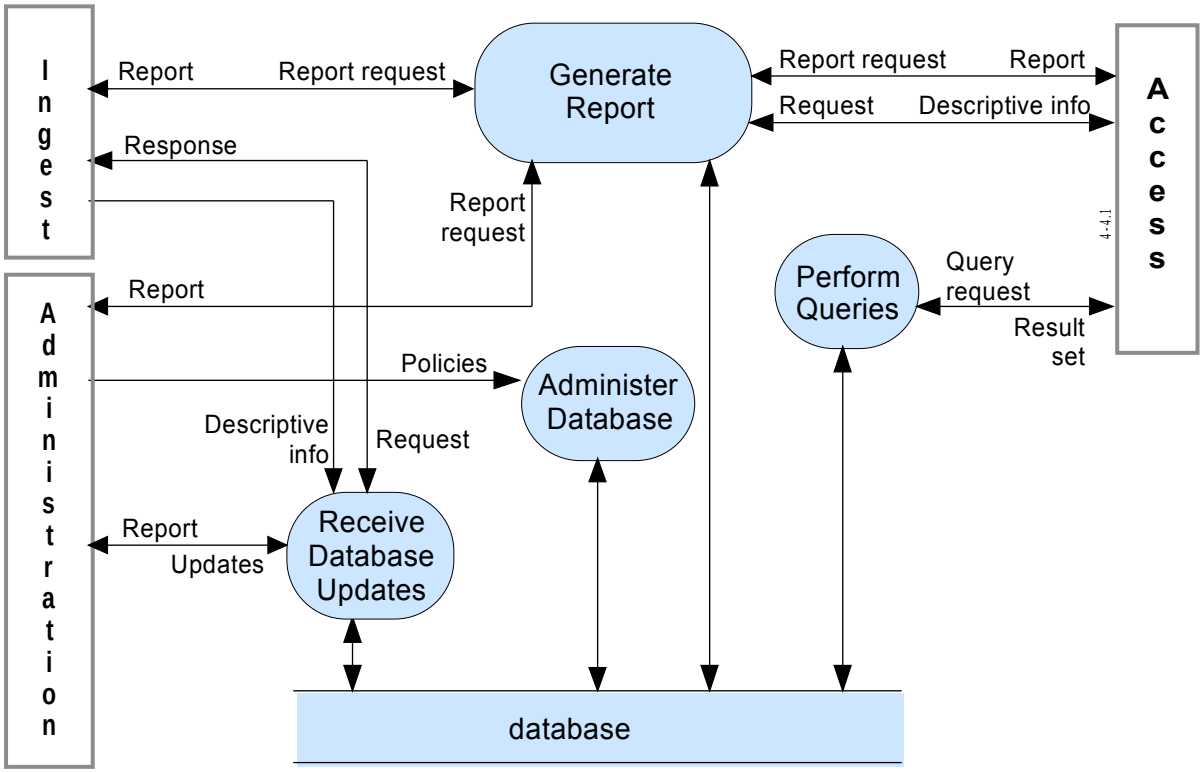


Figure 13: OAIS – Functions of Data Management

4.6.1 Administer Database

4.6.1.1 Administer Database Requirements

The **Administer Database** function is responsible for maintaining the integrity of the Data Management database, which contains both Descriptive Information and system information.

Descriptive Information identifies and describes the archive holdings while system information is used to support archive operations. The Administer Database function is responsible for creating any schema or table definitions required to support Data Management functions; for providing the capability to create, maintain and access customized user views of the contents of this storage; and for providing internal validation (e.g., referential integrity) of the contents of the database. The Administer Database function is carried out in accordance with *policies* received from Administration.

4.6.1.2 Implementation in Rosetta

Rosetta provides a full set of administrative tools/scripts to fully administer the database and supports the following database utilities (partial list):

Number	Utility Name
0/1	**Oracle Server**
0/1/1	Activate Oracle Server
0/1/2	Close Oracle Server
0/1/3	Show Running Oracle Server
0/1/4	Show Oracle Server Status
0/2	**Oracle Listener**
0/2/1	Activate Oracle Listener
0/2/2	Close Oracle Listener
0/2/3	Show Running Oracle Listener
0/2/4	Show Listener Status
0/2/5	Show Listener Services
0/3	**Oracle Logs**
0/3/1	View Oracle ALERTLOG

4.6.2 Perform Queries

4.6.2.1 Perform Queries Requirements

The **Perform Queries** function receives a *query request* from Access and executes the query to generate a *result set* that is transmitted to the requester.

4.6.2.2 Implementation in Rosetta

The 'Manage Sets' workbench provides an interface for staff users to query, access, and create sets containing objects stored in the permanent repository. While material exists in the staging repository, it may be accessed from a variety of contexts, including selection, approval, arrangement, and technical analysis. Those contexts are not available once the material is stored in the permanent repository. The Workbench is the daily interface for authorized staff users that need to handle intellectual entities, representations, files, and metadata records.

The primary Workbench processes include:

- Querying objects
- Creating sets
- Managing sets
- Managing metadata records

The Workbench provides a variety of tools for constructing simple or complex queries for intellectual entities, representations, and files. The results display inline with the query so that the query can be refined as necessary. Several actions are available for handling the query results on an individual basis (i.e., not as a set) from the results list.

The staff user can create sets from the query (accessed from the "Object Search" or "Add Set" wizard) or the results (accessed from the "Add Set" wizard). Logical sets capture and store the query SQL. An object's membership in a logical set is determined at run time, i.e., at the moment the query is executed. Therefore, a logical set's members are dynamic and subject to change each time the query is run. Itemized sets store a list of object ids. An object's membership in an itemized set is determined at the time of saving the set. The itemized set's members are static until a staff user adds members to or removes members from the set.

Both types of sets require a management component. Staff users may update, copy, delete, add to, and perform process automation (available from the Process Automation UI) using sets.

4.6.2.3 Generate Report Requirements

The **Generate Report** function receives a *report request* from Ingest, Access or Administration and executes any queries or other processes necessary to generate the *report* that it supplies to the requester. Typical reports might include summaries of archive holdings by category, or usage statistics for accesses to archive holdings. It may also receive a report request from Access and provides *descriptive information* for a specific AIP.

4.6.2.4 Implementation in Rosetta

Rosetta will provide a set of views that will allow 3rd-party tools to generate reports. These views will be based on multiple sets of tables and allow both count and list type reports. The tables that will be externalized for reports include the following:

- Events
- Statistics
- User Information
- Producers
- Producer Profiles
- Material Flows
- SIP and SIP Item Tracking
- Deposit Activities
- Format Library

4.7 OAIS Administration Module

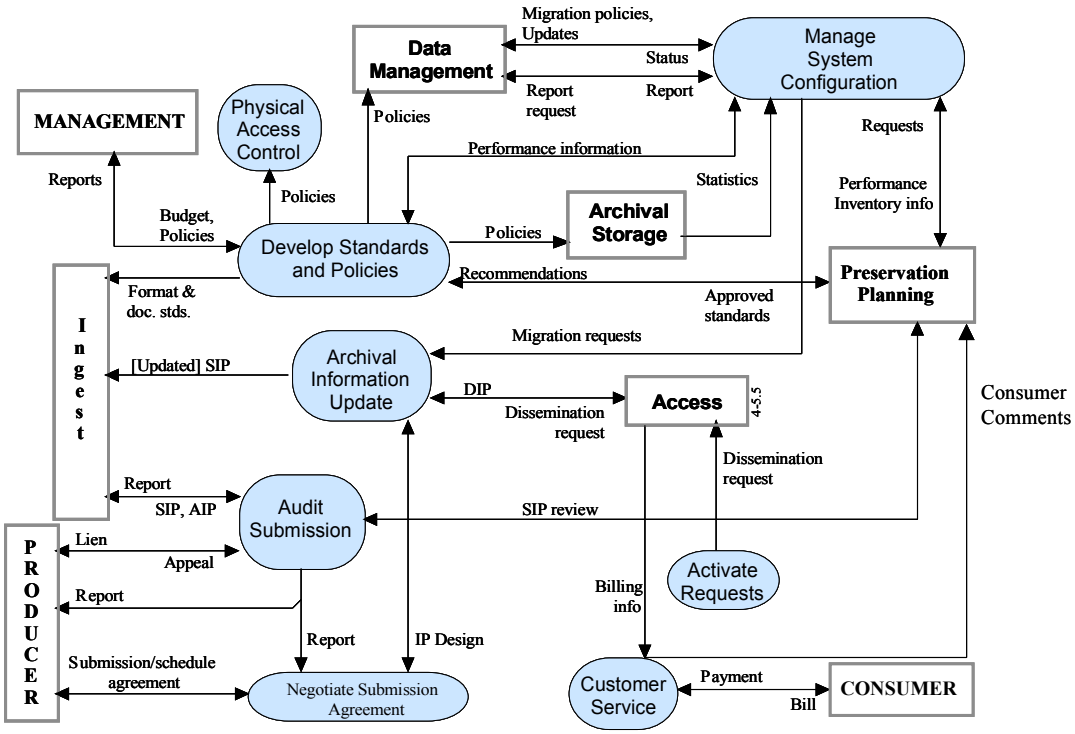


Figure 14 OAIS – Functions of Administration

4.7.1 Manage System Configuration

4.7.1.1 Manage System Configuration Requirements

The Manage System Configuration function provides system engineering for the archive system to continuously monitor the functionality of the entire archive system and systematically control changes to the configuration. This function maintains in-

tegrity and tractability of the configuration during all phases of the system life cycle. It also audits system operations, system performance, and system usage. It sends report requests for system information to Data Management and receives reports; it receives operational statistics from Archival Storage. It summarizes those reports and periodically provides OAIS performance information and archive holding inventory reports to Preservation Planning. It sends performance information to establish standards and policies. It receives migration packages from Preservation Planning. It receives system evolution policies from the Establish Standards and Procedures function. Based on these inputs, it develops and implements plans for system evolution. It sends change requests, procedures, and tools to Archive Information Update.

4.7.1.2 Implementation in Rosetta

Administrators configure the Preservation system in order to define the system infrastructure (such as user accounts, material flows, and content processing instructions) required for using the system. The Preservation system requires Administrators to perform an initial configuration after installation. The initial configuration is performed only once. Later, Administrators can modify the initial configuration settings and perform maintenance tasks on an ongoing basis. The Preservation system supports the following types of configuration:

- **Initial configuration** - Initial configuration entails the configuration of essential components required for using the Preservation system
- **Ongoing configuration** - An Administrator can use ongoing configuration to modify components that were defined during the initial configuration, as well as to perform ongoing system maintenance
- **Advanced configuration** - An Administrator performs advanced configuration in order to define general components that are not frequently changed. Administrators use the Advanced Configuration screen for this purpose.

4.7.2 Archival Information Update

4.7.2.1 Archival Information Update Requirements

The Archival Information Update function provides a mechanism for updating the contents of the archive. It receives change requests, procedures and tools from Manage System Configuration. It provides updates by sending a dissemination request to Access, updating the contents of the resulting DIPs and resubmitting them as SIPs to Ingest.

4.7.2.2 Implementation in Rosetta

“Meditor” is the name of a host of editing and metadata management capabilities that were formerly encapsulated in the standalone Meditor Windows client. In the DPS software, Meditor functions are invoked from the Workbench interfaces by staff users that want to

- Edit object attributes (via the DNX editor)

- Add or edit metadata records
- Perform object services such as: export and run validation stack
- Add representations
- Delete representations
- Upload or download files

4.7.3 Audit Submission

4.7.3.1 Audit Submission Requirements

The Audit Submission function will verify that submissions (SIP or AIP) meet the specifications of the Submission Agreement. This function receives AIP/SIP reviews from Preservation Planning and may also involve an outside committee (e.g., science and technical review). The audit process must verify that the quality of the data meets the requirements of the archive and the review committee. It must verify that there is adequate Representation Information and PDI to ensure that the Content Information is understandable and independently usable to the Designated Community. The formality of the review will vary depending on internal archive policies. The Audit process may determine that some portions of the SIP are not appropriate for inclusion in the archive and must be resubmitted or excluded. An audit report is provided to Ingest. After the audit process is completed, any liens are reported to the Producer, who will then resubmit the SIP to Ingest or appeal the decision to Administration. After the audit is completed, a final ingest report is prepared and provided to the Producer and to Negotiate Submission Agreement. Audit methods potentially include sampling, periodic review, and peer review.

4.7.3.2 Implementation in Rosetta

According to the SIP routing rules, each SIP is assigned to an Approval Group. The Approval group is assigned using the SIP Processing rules. The rule inputs include:

- Material Flow
- Material Type
- Producer Type
- Producer Group
- Producer

Each role of Assessor, Arranger, and Approver is part of an Approval Group. When one member logs in to the Workbench, they can see the SIPs that are assigned to their Approval Group.

Note: A user having all three roles (i.e. Assessor, Arranger, and Approver) will see the relevant SIPs for each stage when entering each Workbench. For example, when a user with both Assessor and Arranger roles enters the Assessor Workbench, he or she sees all the SIPs that are currently assigned to the Assessor. If he or she enters the Arranger Workbench, only SIPs assigned to the Arranger stage and belonging to that Approval Group are present.

The Approval Group value can be overridden only by a privileged user (which has the privilege "Assessor/Arranger/Approver - Full"). The Typical Assessor/Arranger/Approver cannot forward or reassign the SIP to a different group.

All material submitted by Registered Producers that are not Trusted is reviewed manually by Assessors and Re-arrangers to ensure that the SIP's content is appropriate and suitable for preservation.

Deposits from Trusted and Internal Producers are sampled-based on a pre-defined Sampling Rate.

4.8 OAIS Preservation Module

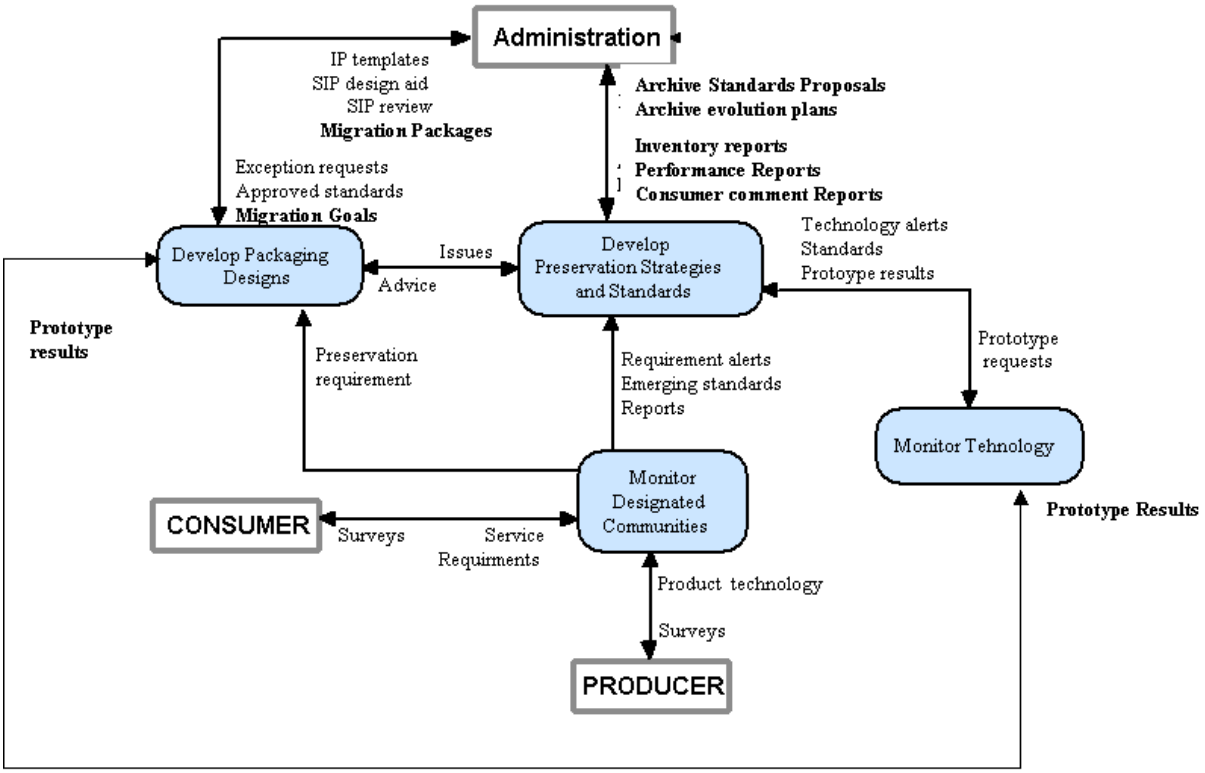


Figure 15: OAIS – Functions of Preservation

4.8.1 Monitor Designated Community

4.8.1.1 Monitor Designated Community requirements

The Monitor Designated Community function interacts with archive Consumers and Producers to track changes in their service requirements and available product technologies. Such requirements might include data formats, media choices, preferences for software packages, new computing platforms, and mechanisms for communicating with the archive. This function may be accomplished via surveys, via a periodic formal review process, via community workshops where feedback is solicited or by individual interactions. It provides reports, requirements alerts and emerging standards to the Develop Preservation Strategies and Standards function. It sends preservation requirements to Develop Packaging Designs.

4.8.1.2 Implementation in Rosetta

The Preservation module is designed to manage the description of the library's technical capability. To do so, a knowledge base (KB) in the form of a set of libraries will be created. This KB will include:

- Format Library
- Application Library
- Classification Group Library
- Risk Identifier Library
- MD Extractors Library

Format Library

Ex Libris will create a Format Library based on the UK's PRONOM database, but extended to:

- support the required risk analysis and preservation planning functionality (creating preservation sets, defining evaluation criteria)
- provide a more complete information store that is tailored to the specific formats represented in the system (local information, sustainability factors, related risks)

The Format Library will link information describing these formats with supporting applications and known risks. It will also support a more detailed description of the relevant format's sustainability factors.

Each institution can have additional information that is managed locally (such as deciding if a risk identifier for a certain format is relevant for its repository). The local information will not be shared with other Rosetta installations.

Application Library

A new Application Library that contains descriptions of known applications will be created. It will support linking these applications with the Format Library. The initial information comes from PRONOM and can be enriched by local information and sustainability factors.

Classification Group Library

A Classification Group Library that contains descriptions of known properties for groups of formats will be created. Because significant properties are usually relevant for groups of formats, sharing descriptions through groups improves efficiency. When creating a preservation plan and defining the evaluation form, planners can use these properties to configure the system to automatically measure the success of the migration.

In addition, the classification group holds a list of MD extractors that can be used by the "members" of the group (formats).

The initial classification group list is based on the following PRONOM categories:

- Image (Raster)

- Image (Vector)
- Audio
- Video
- Database
- Spreadsheet
- Text (Unstructured)
- Text (Structured)
- Text (Mark-up)
- Text (Word-processed)
- Presentation
- GIS
- Page Description
- Email

Since some formats (out of the ~500 formats in PRONOM) do not fall in any of the above categories, there will also be a default classification that includes no risk property fields (F1 – F20).

Risk Identifier Library

The library will contain two kinds of risks:

- Risk extraction tools – These tools are registered in the system through the Plug-in Manager. Each tool is wrapped in a class that is called by the system as part of the risk grading routine (see below). Each tool can be associated with one or more formats, and for each format, a different risk will be assigned to the tool. When creating a risk, the user can choose from a list of registered tools and associate a tool with the risk. (The list contains only tools that are categorized as risk extractors.)

- Property risks – These risks contain conditions of properties and values that put at risk the files that match these conditions. These conditions can be the reason for not being able to render the files, or any other reason that is decided by the institution (for example, files over the size of 1 GB).

Note: The file properties that can be described as risks are configured in the INDEXING_CONFIGURATION table. These fields might be only a sub-set of the significant properties that are defined in the Classification Group library. (For more details about the risk properties see in the Improving Search Capabilities FSD)

The risk report summarizes the counts of files in each format that match these two kinds of risks. For more details regarding risk grading, see the Risk Analysis FSD.

MD Extractors Library

The MD Extractors Library holds the mapping between the output of the MD extractors and the matching DNX records.

The MD extractors are tools that are registered through the Plug-in Manager.

5 Data Model

This section describes the hierarchal structure of the PREMIS data model, how METS is used to support this data model, and what the special significant properties for AV within the data model are.

5.1 PREMIS Compliance

Four levels of objects (as defined by PREMIS) were used as the basis for the Rosetta data model.

Further information related to PREMIS reference model can be found at: http://www.-loc.gov/standards/premis/

The entities in the data model are defined as follows:

5.1.1 Intellectual Entity

A set of content that is considered a single intellectual unit for purposes of management and description: for example, a particular book, map, photograph, or database. An Intellectual Entity may have one or more digital representations.

5.1.2 Representations

The goal of many preservation repositories is to maintain usable versions of intellectual entities over time. For an intellectual entity to be displayed, played, or otherwise made usable to a human, all of the files making up at least one version of that intellectual entity must be identified, stored, and maintained so that they can be assembled and rendered to a user at any given point.

A representation is the set of files required to do this.

A representation is a single digital instance of an intellectual entity held in the repository.

A preservation repository might hold more than one representation for the same intellectual entity.

5.1.3 Files

A file in the PREMIS data model is similar to the idea of a computer file in ordinary usage: a set of zero or more bytes known to an operating system. Files can be read, written, and copied. Files have names and formats.

5.1.4 Bitstreams

A bitstream as defined in the PREMIS data model is a set of bits embedded within a file. This differs from common usage, where a bitstream could in theory span more than one file.

A good example of a file with embedded bitstreams is a TIFF file containing two images.

According to the TIFF file format specification, a TIFF file must contain a header containing some information about the file. It may then contain one or more images. In the

data model each of these images is a bitstream and can have properties such as identifiers, location, inhibitors, and detailed technical metadata (e.g., colour space).

Some bitstreams have the same properties as files and some do not. The image embedded within the TIFF file clearly has properties different from the file itself. However, in another example, three TIFF files could be aggregated within a larger tar file. In this case, the three TIFF files are also embedded bitstreams, but they have all the properties of TIFF files.

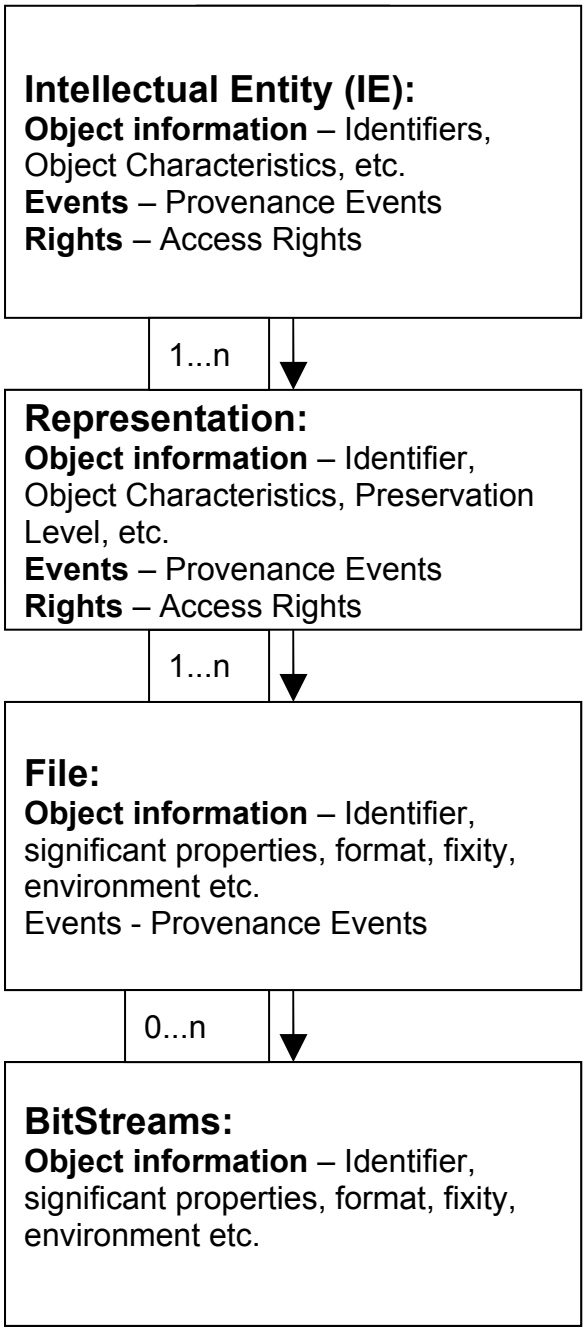


Figure 16: PREMIS Data Model

5.2 METS - Metadata Encoding and Transmission Standard (METS)

Further information related to METS reference model can be found at: http://www.-loc.gov/standards/mets/

Rosetta is using METS as a container for the SIP/AIP which are compliant with the PREMIS structural model.

The METS schema contains 3 types of metadata: Descriptive, Administrative, and Structural Map.

The following table illustrates which MD type is applicable to which object type:

	Descriptive	Administrative	Structural Map
IE	√	√	
REP		√	√
FILE		√	
BitStream		√	

Descriptive: information relating to the intellectual contents of the object, akin to much of the content of a standard catalogue record. This enables the user of a digital library to find the object and assess its relevance. Rosetta supports DC (Dublin Core) as the standard for descriptive metadata.

Administrative: information necessary for the manager of the electronic collection to administer the object, including information on intellectual property rights and technical information on the object and the files that comprise it.

Structural: information on how the individual components that make up the object relate to each other, including the order in which they should be presented to the user (for example, how still-image files that comprise a digitized version of a print volume should be ordered).

5.2.1 METS Sections

A METS file consists of seven major sections, each describing a different facet of the digital object.

5.2.1.1 Header

The METS Header contains metadata describing the METS document itself, including such information as creator, and editor.

The Header section is not in use in Rosetta and is not included in the Rosetta data model.

5.2.1.2 Descriptive Metadata

Limitations:

- Only on IE level – Since the IE is "a set of content that is considered a single intellectual unit for purposes of management and description (PREMIS)", Rosetta limits the descriptive metadata for the IE level only
- Descriptive metadata is currently limited to Dublin Core (DC) and Qualified DC standards

Descriptive metadata for an IE, is held within a section of the METS file named <dmdSec>. Each <dmdSec> is labeled with a unique ID to allow it to be referenced from within the structural map. METS allows this metadata either to be held in external files which are referenced from within the METS file, or to be embedded directly with it.

The descriptive MD in Rosetta is embedded directly in the METS file, using an <mdWrap> element to contain it:

```
<mets:dmdSec ID="ie-dmd">
    <mets:mdWrap MDTYPE="DC">
      <mets:xmlData>
        <dc:record xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <dc:creator>Mark Twain</dc:creator>
          <dc:identifier>ISBN 90-70002-34-5</dc:identifier>
          <dc:date/>
          <dc:publisher/>
          <dc:description/>
          <dc:title>Tom Sawyer</dc:title>
        </dc:record>
      </mets:xmlData>
    </mets:mdWrap>
  </mets:dmdSec>
```

5.2.1.3 Administrative Metadata

Administrative metadata, including technical information on the digital files that comprise an object, and information on intellectual property rights attached to it, is dealt with in a manner analogous to descriptive metadata. <amdSec> elements, each identified by an ID, are used to record this metadata, which may be held in external files or embedded within the METS file using the <mdWrap> element. Rosetta uses DNX format for holding all of the following elements that are related to each object:

- technical (mets:techMD)
- rights (mets:rightsMD)
- source (mets:sourceMD)

- events (mets:digiprovMD)

Example of techMD section of a Representation:

```
<mets:amdSec ID="REP1001-amd">
    <mets:techMD ID="REP1001-amd-tech">
      <mets:mdWrap MDTYPE="OTHER" OTHERMDTYPE="dnx">
        <mets:xmlData>
          <dnx xmlns="http://www.exlibrisgroup.com/dps/dnx">
            <section id="generalRepCharacteristics">
              <record>
                <key id="preservationType">PRESERVATION_MASTER</key>
                <key id="usageType">VIEW</key>
                <key id="RevisionNumber">1</key>
                <key id="DigitalOriginal">false</key>
              </record>
            </section>
            <section id="internalIdentifier">
              <record>
                <key id="internalIdentifierType">PID</key>
                <key id="internalIdentifierValue">REP1001</key>
              </record>
            </section>
            <section id="objectCharacteristics">
              <record>
                <key id="objectType">REPRESENTATION</key>
                <key id="creationDate">2009-05-31 17:50:29</key>
                <key id="createdBy">admin1</key>
                <key id="modificationDate">2009-05-31 17:50:29</key>
                <key id="modifiedBy">admin1</key>
                <key id="owner">CRS00.INS00.DPR00</key>
              </record>
            </section>
          </dnx>
        </mets:xmlData>
      </mets:mdWrap>
    </mets:techMD>
```

Example of rightsMD of the same Representation:

```
    <mets:rightsMD ID="REP1001-amd-rights">
      <mets:mdWrap MDTYPE="OTHER" OTHERMDTYPE="dnx">
```

```
        <mets:xmlData>
          <dnx xmlns="http://www.exlibrisgroup.com/dps/dnx"/>
        </mets:xmlData>
      </mets:mdWrap>
    </mets:rightsMD>
```

Example of sourceMD section: The source MD is the MD that came with the object into the repository and it was not generated by Rosetta. This MD is referenced from within the METS file and it is held in the DB.

```
<mets:sourceMD ID="REP1001-amd-source">
      <mets:mdWrap MDTYPE="OTHER" OTHERMDTYPE="dnx">
        <mets:xmlData>
          <dnx xmlns="http://www.exlibrisgroup.com/dps/dnx">
            <section id="metaData">
              <record>
                <key id="MID">DNX_REP1001</key>
                <key id="UUID">20806</key>
                <key id="creationDate">2009-05-31 17:50:29</key>
                <key id="createdBy">admin1</key>
                <key id="modificationDate">2009-05-31 17:50:30</key>
                <key id="modifiedBy"/>
                <key id="metadataType">21</key>
                <key id="description"/>
                <key id="externalSystem"/>
                <key id="externalRecordId"/>
              </record>
            </section>
          </dnx>
        </mets:xmlData>
      </mets:mdWrap>
    </mets:sourceMD>
```

The following is an example of digiProvMD section of a file. This section holds the events that are stored in the METS (Provenance Events).

```
<mets:digiprovMD ID="FL1002-amd-digiprov">
      <mets:mdWrap MDTYPE="OTHER" OTHERMDTYPE="dnx">
        <mets:xmlData>
          <dnx xmlns="http://www.exlibrisgroup.com/dps/dnx">
            <section id="event">
              <record>
                <key id="eventDateTime">2009-05-31 17:50:36</key>
```

```
                <key id="eventType">VALIDATION</key>
                <key id="eventIdentifierType">DPS</key>
                <key id="eventIdentifierValue">27</key>
                <key id="eventOutcome1">SUCCESS</key>
                <key
id="eventOutcomeDetail1">PID=FL1002;PRODUCER_ID=10000;MF_ID=1;ALGORITHM_NA
ME=SHA1;DEPOSIT_ACTIVITY_ID=1;SIP_ID=1;</key>
                <key id="linkingAgentIdentifierType1">SOFTWARE</key>
                <key id="linkingAgentIdentifierValue1">REG_SA_JAVA5_FIX-
ITY</key>
              </record>
            </section>
          </dnx>
        </mets:xmlData>
      </mets:mdWrap>
    </mets:digiprovMD>
  </mets:amdSec>
```

5.2.1.4 File Groups

The individual files of which the digital object as a whole is comprised are listed in the file group section. Each file is referenced by a file element, such as:

```
  <mets:fileSec>
    <mets:fileGrp USE="VIEW" ID="REP1314" ADMID="REP1314-amd">
      <mets:file ID="FL1317" MIMETYPE="image/tiff" ADMID="FL1317-amd">
        <mets:FLocat LOCTYPE="URL" xlin:href="/exlibris/dps/ file_1/V1-
FL1317.tif" xmlns:xlin="http://www.w3.org/1999/xlink"/>
      </mets:file>
      <mets:file ID="FL1316" MIMETYPE="image/tiff" ADMID="FL1316-amd">
        <mets:FLocat LOCTYPE="URL" xlin:href="/exlibris/dps/file_1/V1-
FL1316.tif" xmlns:xlin="http://www.w3.org/1999/xlink"/>
      </mets:file>
      <mets:file ID="FL1315" MIMETYPE="image/tiff" ADMID="FL1315-amd">
        <mets:FLocat LOCTYPE="URL" xlin:href="/exlibris/dps/file_1/V1-
FL1315.tif" xmlns:xlin="http://www.w3.org/1999/xlink"/>
      </mets:file>
    </mets:fileGrp>
  </mets:fileSec>
```

The physical location of the file, usually given in the form of a URL, is indicated by the <Flocat> element, while its ID, by which it is referenced from the structural map, is declared by the ID attribute of the <file> element itself.

5.2.1.5 Structural Map

This key part of a METS file, is a description of the overall structure of the object (its structural metadata). This section describes the major components within the ob-

ject, and how they relate to each other hierarchically. For example, if the object is a digitized book, this section will show that the book as a whole is divided into separate chapters, and if these chapters themselves contain sections or subsections, it the structural description will show how these are nested together.

A very simple structural map for a book may look like this:

```
<mets:structMap ID="REP1314-1" TYPE="PHYSICAL">
  <mets:div LABEL="PRESERVATION_MASTER;VIEW">
    <mets:div LABEL="Table of Contents">
      <mets:fptr FILEID="FL1315"/>
      <mets:fptr FILEID="FL1316"/>
      <mets:fptr FILEID="FL1317"/>
    </mets:div>
  </mets:div>
</mets:structMap>
```

This structural map shows that the Representation is divided into 3 pages, and that the first page is divided into two files. Within these sections are pointers to the files that hold the images of the pages: these are referenced by the FILEID attribute within the <`fptr`> (file pointer) element.

The structural map provides a logical layout for the structure of the whole object, and one that is easy to navigate using any XML-compatible software. In the Delivery module in Rosetta, the viewer displays the files according to this order:

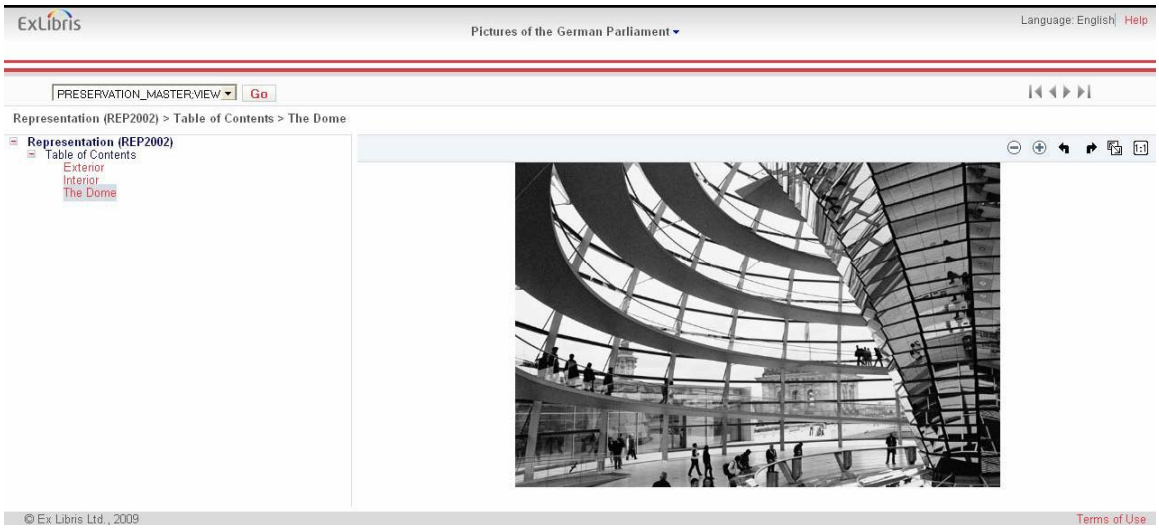


Figure 17: PREMIS Data Model

5.2.1.6 Structural Links

Not in use in Rosetta data model.

5.2.1.7 Behaviour

Not in use in Rosetta data model.

5.3 DNX – Digital-preservation Normalized XML

The motivation for creating the DNX is to have a simple and unified XML schema that will contain all the important data elements in a simple flat structure, contains the same structure according to the PREMIS data model and including the important technical metadata that is relevant for Preservation.

The DNX contains all the PREMIS semantic units and other data elements for holding all the relevant metadata in a normalized way.

Unlike PREMIS hierarchic structure, the DNX is much more 'flat' and uses a 'key/value' structure.

The DNX section is implementing it as follows, where the PREMIS **container** is called **section** and the **semantic components** are called **records** that include **keys**:

```
<section id="internalIdentifier">
              <record>
                <key id="internalIdentifierType">SIPID</key>
                <key id="internalIdentifierValue">1</key>
              </record>
              <record>
                <key id="internalIdentifierType">PID</key>
                <key id="internalIdentifierValue">BS1</key>
              </record>
              <record>
                <key id="internalIdentifierType">DepositSetID</key>
                <key id="internalIdentifierValue">1</key>
              </record>
            </section>
```

5.3.1 DNX Sections

The DNX is built with sections where each section gathers common elements. Examples for some of the DNX sections:

- generalIECharacteristics – Gathers IE elements such as: creation date, modification date, etc
- generalRepCharacteristics – Gathers REP elements such as: Preservation type, creation date, modification date, etc
- CMS (Collection Management System) – Gathers elements related to CMS such as system and recordID.

Gathering the different elements into sections makes the DNX much more usable.

5.3.2 Significant Properties Section

The most dominant and meaningful DNX section from a preservation plan/action perspective, is the 'significant properties' section. It contains the validation tool (JHOVE, NLNZ or any other plug-in tool) output such as height, width, bit depth, and bit rate.

Significant properties may be objective technical characteristics subjectively considered important or subjectively determined characteristics. For example, a PDF may contain links that are not considered important and JavaScript that is considered important.

Or future migrations of a TIFF image may require optimization for line clarity or for color; the option chosen would depend upon a curatorial judgment of the significant properties of the image.

5.4 Events

The Event entity aggregates metadata about actions. A preservation repository will record events for many reasons. Documentation of actions that modify (that is, create a new version of) a digital object is critical to maintaining digital provenance, a key element of authenticity. Actions that create new relationships or alter existing relationships are important in explaining those relationships. Even actions that alter nothing, such as validity and integrity checks on objects, can be important to record for management purposes. For billing or reporting purposes some repositories may track actions such as requests for dissemination or reports.

Rosetta maintain provenance event for each of the object levels. Each such event will be written in the events (`mets:digiprovMD`) section belongs to the relevant object level.

Each event will be written in a DNX format and will include the following:

- Agent – The agent who triggered this event. Agent is not necessarily a person. Agent may refer also to a process, plug-in tool, etc.
- Event details – such as: Creation date, description, parameters, etc.

The following is an example of the `digiProvMD` section of a file. This section holds the events that are stored in the METS (Provenance Events).

```
<mets:digiprovMD ID="FL1002-amd-digiprov">
      <mets:mdWrap MDTYPE="OTHER" OTHERMDTYPE="dnx">
        <mets:xmlData>
          <dnx xmlns="http://www.exlibrisgroup.com/dps/dnx">
            <section id="event">
              <record>
                <key id="eventDateTime">2009-05-31 17:50:36</key>
                <key id="eventType">VALIDATION</key>
                <key id="eventIdentifierType">DPS</key>
                <key id="eventIdentifierValue">27</key>
                <key id="eventOutcome1">SUCCESS</key>
                <key
id="eventOutcomeDetail1">PID=FL1002;PRODUCER_ID=10000;MF_ID=1;ALGORITHM_NAME=S
HA1;DEPOSIT_ACTIVITY_ID=1;SIP_ID=1;</key>
                <key id="linkingAgentIdentifierType1">SOFTWARE</key>
                <key
id="linkingAgentIdentifierValue1">REG_SA_JAVA5_FIXITY</key>
              </record>
```

```
        </section>
      </dnx>
    </mets:xmlData>
  </mets:mdWrap>
</mets:digiprovMD>
```

In addition to the above mentioned events, Rosetta will keep the producer and producer agent information at the digiprovMD section of an IE.

```
<mets:digiprovMD ID="ie-amd-digiprov">
  <mets:mdWrap MDTYPE="OTHER" OTHERMDTYPE="dnx">
    <mets:xmlData>
      <dnx xmlns="http://www.exlibrisgroup.com/dps/dnx">
        <section id="producer">
          <record>
            <key id="userName"/>
            <key id="address1"/>
            <key id="address2"/>
            <key id="address3"/>
            <key id="address4">Afghanistan</key>
            <key id="address5">New Zealand</key>
            <key id="defaultLanguage">en</key>
            <key id="emailAddress">zvi.fass@exlibris.co.il</key>
            <key id="firstName">INS00</key>
            <key id="jobTitle"/>
            <key id="lastName">gfghf</key>
            <key id="middleName"/>
            <key id="telephone1">34423423</key>
            <key id="telephone2"/>
            <key id="authorativeName">Demo Internal Producer</key>
            <key id="producerId">10000</key>
            <key id="userIdAppId">10000</key>
            <key id="webSiteUrl"/>
            <key id="zip"/>
          </record>
        </section>
        <section id="producerAgent">
          <record>
            <key id="firstName">Abie</key>
            <key id="lastName">Ackart</key>
            <key id="middleName"/>
          </record>
```

```
            </section>
          </dnx>
        </mets:xmlData>
      </mets:mdWrap>
    </mets:digiprovMD>
```

5.5 Access Rights

Two different access rights types exist:

- PREMIS – PREMIS defines the rights information regarding an external system that manages the access rights.

- linkingRightsStatementIdentifierType - A designation of the domain within which the linkingRightsStatementIdentifier is unique

- linkingRightsStatementIdentifierValue - The value of the linkingRightsStatementIdentifier

- Rosetta – Rosetta defines the rights information related to what the institution is allowed to do with the object. This information includes:

- PolicyID – The unique ID of the different access rights managed by Rosetta. For example: AR_EMBARGOED_FOR_5_YEARS, AR_5_CONCURRENT_USERS, etc

- Policy parameters – If the policy requires any parameters

- Policy description – Description of the policyID. For example:

1. AR_EMBARGOED_FOR_5_YEARS - Embargoed for 5 years

2. AR_5_CONCURRENT_USERS - Limited access according to copyright law

6 Terms

Term	Description
Access Rights	Define the access rights for an object, if one can view an object or not.
Access Rights Checker	A Plugable Component that determines access restrictions for a user, from a particular context, for a particular object
Access SDK	A code component that externalizes all the relevant APIs needed for creation of Viewers and Viewer Pre Processors
Application Library	The Application Library contains all of the data regarding applications: name, ID, license end-date, and so forth. Each application can be related to one or more formats. The Application Library is managed globally (exposed to all installations of Rosetta) and some information is managed in each installation. The connections between the Format Library and the Application Library are managed locally, but new applications (as new formats) should be added or removed in the global libraries.
Bitstream	An object embedded within a ByteStream that cannot be transformed into a standalone file without the addition of file structure (e.g., headers).
ByteStream	Compound file containing FileStream/s and/or embedded BitStreams
Classification Group	Since significant properties are usually shared between multiple formats, the classification group is a way to aggregate these common properties so that they will be connected to all the relevant formats.
DNX	DNX metadata is implemented as an XML record, containing by unlimited list of sections (or "records") where each record contains unlimited number of attributes (or "properties"). Neither the "record" names not the attributes names are limited by definition so in theory can hold any information. In practice, we will introduce DNX configurable profile, each limited this flexibility and enables configurable validation, ReadOnly section and default values. The motivation for defining yet another md-type called DNX came from the need to collapse and aggregate administrative and technical metadata onto one "roof" where the management, development and viewing/editing is much more simple.
DPS SDK	Set of programming tools that allow programmers to develop specialized computer applications and adapt them to DPS.
Event	An Event is an action that involves at least one entity or agent in the system
File	A file is a named and ordered sequence of bytes that is known by an operating system. One or more files compose any given Representation
FileStream	True file embedded within a larger compound file (ByteStream). A FileStream has all of the properties of a ByteStream.
Format Library	The Format Library contains all of the data regarding formats: name, description, related applications, related risks, and sustainability factors. Some of the information is managed globally (exposed to all installations of Rosetta) and some of it can be managed locally (stored within the local DB of the institution). In the local library, data can be added but not removed.
Human Stage	a stage performed by a human agent (e.g., Technical Analyst, Assessor, Approver) who uses the application's UI to manipulate the processed SIP. Upon entering a Human Stage, the application will forward the SIP's information to the appropriate agents and will mark the SIP as waiting for respond. The SIP automatic processing will continue once the appropriate human agent will be done working on the SIP.
Intellectual Entity	Is a coherent set of content that is reasonably described as a unit, for example, a particular book, map, photograph, or database. An Intellectual Entity may have one or more Representations.
Itemized Set	A group of object whose set members are determined at the time the set is saved. There is no stored relation between the set members and the query from which they were derived.

JBPM	Stands for Java Business Processes Manager. The JBPM is a Workflow and BPM engine that enables the creation and management of business processes that execute the Stage Routines and keep tracking of the SIP during its processing.
Logical Set	A group of objects whose set members are determined at run time; i.e., when the set's stored query is executed
Material Flow	A set of configuration data that both defines a Web interface made up of a sequence of forms for submitting digital material through the Deposit Application as well as instructions on how to process the submitted data for loading into the Staging server. A given institution can define as many Material Flows as needed. A Material Flow is associated with one or many *Producers* / Producer Profiles.
Material Flow configuration	A set of configuration rules and processes (Metadata forms, file-acquiring methods, copyright information) implemented by the deposit application to generate the required Material Flow screens.
Meditor	The DPS interface through which intellectual entities, representations, and files are edited
Metadata Form	A form used by the user to fill in metadata information regarding a deposited object
Metadata Management	A component of the Workbench that allows users to search, view, and edit metadata records as discrete objects; i.e., devoid of their association to any repository intellectual entity, representation, or file
METS	Stands for Metadata Encoding and Transmission Standard, METS is a common and widely use format. See: http://www.loc.gov/standards/mets/. The DPS staging server receives the data encoded in METS format. Deposit Application is a DPS application used by producer agents to deposit digital materials. The deposit application is responsible for the file(s) acquisition and for the conversion from various Content Structures to a METS format. The Deposit Application is constructed from two main layers: the Deposit API layer and the Deposit Web interface.
Migration types	At this point the only type of alternative supported in Rosetta is Migration. Migration means converting files from one format to a different one. The alternative could be either internal or external: Internal - The converting tool is registered in Rosetta and it is being activated as part of the process automation framework External – The actual conversion is done outside of Rosetta and the preservation execution starts with receiving the converted files.
Negotiator	Producers, Producer Profiles, and Material Flows are associated with one Institution. In order to allow subdivisions of such entities into more manageable sets, the notion of a Producer Group is introduced. Negotiators assigned a specific set of Producer Groups can manage only those Producers and associated entities tagged with the same Producer Groups.
OAI-PMH	The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a low-barrier mechanism for repository interoperability. Data Providers are repositories that expose structured metadata via OAI-PMH. Service Providers then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP
OAIS	An Open Archival Information System (or OAIS) is an archive, consisting of an organization of people and systems, that has accepted the responsibility to preserve information and make it available for a Designated Community. OAIS is the ISO reference model for Open Archival Information System.
OTB	Out of the box, this term stands for a set of configurations that is part of the installation, meaning, this configuration will be available at first time.

PDS	This stands for Patron Directory Service and is a “back-end” DPS Web component that facilitates user authentication and login in to the DPS. The PDS is part of the standard calling application package, but it is a distinct and separate component. It does not have a user database of its own. Rather, it can be configured to work against the institution’s authentication server(s) and user database(s), such as an LDAP directory service
Permanent Repository	This is the final storage location for all objects processed and submitted to the system. It provides services such as Delivery and Publishing and information stored in the objects serves to enable efficient and correct preservation risk analysis and actions.
PREMIS	Stands for: "Preservation Metadata: Implementation Strategies". This is an international working group charged with the following: • define an implementable set of “core” preservation metadata elements, with broad applicability within the digital preservation community; • draft a Data Dictionary to support the core preservation metadata element set; • examine and evaluate alternative strategies for the encoding, storage, and management of preservation metadata within a digital preservation system, as well as for the exchange of preservation metadata among systems; • conduct pilot programs for testing the group’s recommendations and best practices in a variety of systems settings; and • explore opportunities for the cooperative creation and sharing of preservation metadata
Preservation Plan	The preservation plan is a structured workflow used by the Preservation Analyst (PA) to handle objects that are at risk. The workflow takes the PA through the stages of gathering documentation and general information, creating the preservation set, defining the suggested alternatives, running tests, and summarizing the results.
Preservation Plan Alternative	Each preservation plan may have more than one alternative in order to ensure success. For example, the same plan may have two migration utilities that convert the source format to the target format. Each one of these is saved as an alternative, and the workflow allows the user to evaluate each utility and add information that is relevant for the utility being evaluated.
Preservation Plan Execution	After the institution signs off on the plan, the plan can be executed with no need to go through testing and defining the exact material. The plan’s execution can be scheduled in advance or launched immediately
Preservation Set	A preservation set is a set of intellectual entities (IEs) that is defined as the first stage of the preservation planning. The set starts as a logical set and becomes an itemized set every time a preservation plan is executed
Pre-Transformer	This is a routine that converts non standard content structure in standard content structure, such that it can be transformed using of the system's standard transformers. It is activated prior to the transformer as part of the SIP Submission process.
Privileges	The discrete permissions that make up a user role. Privileges correspond to the interfaces and interface functions in the DPS.
Producer	An entity (person or organization) in whose name the digital material is submitted to the digital archive. A Producer is associated with one or many Producer Agents. Material flows and deposit parameters such as disk space quota and target group are assigned on the Producer level and apply to all associated Agents.

Producer Agent	A user who deposits digital material for the repository. Agents are always associated with one (and only one) Producer. A user may be associated with more than one Producer Agent role (typically staff depositing for the library as well as on behalf of a Producer).
Provenance	The documentation of the chain of events and actions (as well as related agents) that a specific object has undergone in the Repository.
Provenance Event	A Provenance Event is an action that involves at least one object or agent known to the Repository
Publishing	An extensible process that extracts and formats metadata for external uses
Representation	This is the set of files, including structural metadata, needed for a complete and reasonable rendition of an Intellectual Entity.
Risk Identifiers	Risk can be either a query on the file attributes that put the file at risk (existing technical metadata) or a tool that extracts the technical metadata that describes the problem. Each format can be related to one or more risk identifiers.
Set	A physical list of objects. Sets could be created in two ways. The first way, leading to the creation of an Itemized set, is for the user or system to specifically select (“handpick”) the objects that will be included in the set. Created this way, the set is a collection of selected items. The other way is for the user to define the set in SQL terms (using either dynamic or fixed SQL statement) and let the system generate the set on run-time. Created this way, the set is a collection of objects that matched the Set’s SQL criteria at the moment the set was created.
Set Management	A component of the Workbench that allows authorized staff users to create and administer logical sets and itemized sets
Set Member	An intellectual entity, representation, or file that belongs to a logical set or itemized set.
SIP Processing	This is the process undergone by the SIP from the time it is received by the Staging Server until it is moved to the Permanent Repository. The SIP processing resembles an assembly line: the SIP is automatically moved between processing stages, according to the processing configuration that applies to the SIP. In each stage, the SIP is processed according to the stage's predefined processing instructions. The final stage in the processing of a SIP that completed processing successfully would be the move to the Permanent Repository.
SIP Processing Configuration	stands for a specific set of stages, rules of flow between the stages, and processing instructions that is managed in the JBPM. Processing Configuration may vary between different types of SIPs, according to the predefined Routing Rules. When a SIP enters the Staging Server, the SIP Routing Officer decides, based on the predefined SIP Routing Rules, which SIP Processing Configuration applies to the SIP. Once the relevant SIP Processing Configuration has been identified, the SIP is processed according to the instructions found in the relevant configuration.
SIP Processing Workflow	This is the process that handles the SIP from the point of submission to the Staging Server to the point that it moves to the Permanent Repository. The SIP Processing Workflow includes the following phases: - Transform the METS file to AIP. - Validation Stack. - Approval. - Enrichment. - Move to Permanent Repository

SIP Processing Workflow Configuration	This is a specific set of stages, rules of flow between the stages, and processing instructions. Processing Configuration may vary between different types of SIPs, according to the predefined Routing Rules. When a SIP enters the Staging Server, the SIP Routing Officer decides, based on the predefined SIP Routing Rules, which SIP Processing Configuration applies to the SIP. Once the relevant SIP Processing Configuration has been identified (and linked to the SIP's entry in the SIP Tracking Table), the SIP is processed according to the instructions found in the relevant configuration. The SIP Processing State Machine uses the relevant SIP Processing Configuration as a manual describing how the specific SIP should be handled throughout its processing
SIP Routing Officer	This is a component that identifies, based on the predefined SIP Routing Rules, the SIP Processing Configuration that applies to a specific SIP
SIP Routing Rules	This is a set of parameter-based rules that point out the appropriate SIP Processing Configuration for a combination of SIP attribute values
SIP Tracking Table	Contains information about SIPs in the Staging Server. Used for controlling the processing of SIPs. Each entry in the table relates to a single SIP and contains information such as the SIP's current stage, the status of the stage , the name of the Stage Routine that should be executed, the id of the SIP worker that currently handles this SIP, etc. The SIP Tracking Table is controlled by the SIP Tracking Table Manager. The SIP Tracking Table is a subset of the SIP Registry Table
SIP Tracking Table Manager	This is a component that acts as the gatekeeper of the SIP Tracking Table.
Staging Server	The Staging Server is a DPS application server that receives and processes the SIP. Responsibilities include: - Transform the SIP METS file to AIPs. - Load the AIPs to the staging server. - Run Validation stack. - Execute the approval workflow. - Enrich the submitted data. (create additional manifestations) - Transfer the AIPs to the Permanent Repository.
Statistics Event	This is an Event which is a calculated aggregate of events over a period of time, used for determining event measures (average, number, etc..)
Structural Map	Deposited METS can contain Structural Maps for the entire IE, a group of Representations, or a single Representation. A Structural Map for the entire IE is a Structural Map within the deposited METS that does not reference the Id of any of the METS' file groups. Such a Structural Map will be linked to the Intellectual Entity, will be preserved and can be exported. However, delivery and other operations may not be available for such a Structural Map. A Structural Map for a single Representation is a Structural Map within the deposited METS that shares an Id with a single fileGrp. A Structural Map for multiple Representations (i.e., shared Structural Map) is a Structural Map within the deposited METS that references the Ids of several file groups within the METS. The syntax for multiple references is: ID="fileGrp_1 Id ; fileGrp_2 Id; … " In the case of a shared Structural Map, the Staging Server will be responsible for creating a copy of the Structural Map for each of the relevant Representations upon loading the SIP to the Staging Server (since in the Repository, every Representation is self-contained and contains its Structural Map).
Submission Information Package (SIP)	Contains information about provenance, location of the Submission Content, and the Content Structure. The DPS system defines an XML representation of a SIP (based on METS). The Staging Server receives an XML representation of a SIP from the deposit application, and handles it.

Task	Is a program that performs and operation, within the Core Repository, on an object.
TaskChain	An ordered list of tasks
Transformer	This is a program which converts standard Content Structure into an SIP METS format.
User	An individual or organization that interacts in some way with the system. A user may be a staff member or a patron that logs in to a module and uses the system or a user may be an organization in a more general sense. A user of the system can be assigned various roles such as Negotiator, Approver, and/or Technical Analyst. Some roles are limited to staff users, whereas others such as Producer Agent can be assigned both to staff users and patrons.
User Role	A named group of privileges that an operator is authorized to perform. Roles are based on expected workflows and job responsibilities in the DPS. They are fixed and not editable by the library.
Viewer	An extensible component that handles the viewing of content.
Viewer Pre Processor	An extensible component that is activated based on the delivery rules to prepare an object for viewing and redirect user to the relevant viewer