Abstract: This document describes the concepts underlying the main components supporting the access to an OAIS archive.
This document describes the CASPAR OAIS-based access model, in all its components: authenticity, DRM, persistent identifiers and discovery.

Approval
David Giaretta

Summary
Information access, authenticity, DRM, persistent identifiers, information discovery

Availability PUBLIC

Document Status Sheet

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Comment</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>0_1</td>
<td>11 December 2007</td>
<td>Document skeleton created</td>
<td>Carlo Meghini (editor)</td>
</tr>
<tr>
<td>1_0</td>
<td>9 February 2008</td>
<td>Final draft created</td>
<td>Carlo Meghini (editor)</td>
</tr>
<tr>
<td>1_1</td>
<td>15 February 2008</td>
<td>Slight reformatting and improving consistency.</td>
<td>David Giaretta</td>
</tr>
<tr>
<td>1_2</td>
<td>20 Feb 2008</td>
<td>Small wording changes to wording about Authenticity</td>
<td>Mariella Guercio</td>
</tr>
<tr>
<td>1_3</td>
<td>22 Feb 2008</td>
<td>Updated text on Identifiers</td>
<td>Steve Crothers and David Giaretta</td>
</tr>
</tbody>
</table>
### Project information

<table>
<thead>
<tr>
<th>Project acronym:</th>
<th>CASPAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project full title:</td>
<td>Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval</td>
</tr>
<tr>
<td>Proposal/Contract no.:</td>
<td>IST-2006-033572</td>
</tr>
</tbody>
</table>

### Project Officer: Carlos Oliveira

<table>
<thead>
<tr>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFSO-E3</td>
</tr>
<tr>
<td>Information Society and Media Directorate General</td>
</tr>
<tr>
<td>Content - Learning and Cultural Heritage</td>
</tr>
<tr>
<td>Postal mail:</td>
</tr>
<tr>
<td>Bâtiment Jean Monnet (EUFO 1167)</td>
</tr>
<tr>
<td>Rue Alcide De Gasperi / L-2920 Luxembourg</td>
</tr>
<tr>
<td>Office address:</td>
</tr>
<tr>
<td>EUROFORUM Building - EUFO 1167</td>
</tr>
<tr>
<td>10, rue Robert Stumper / L-2557 Gasperich / Luxembourg</td>
</tr>
<tr>
<td>Phone:</td>
</tr>
<tr>
<td>Fax:</td>
</tr>
<tr>
<td>Mobile:</td>
</tr>
<tr>
<td>E-mail: <a href="mailto:Carlos.Oliveira@ec.europa.eu">Carlos.Oliveira@ec.europa.eu</a></td>
</tr>
</tbody>
</table>

### Project Co-ordinator: David Giaretta

<table>
<thead>
<tr>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>STFC (formerly CCLRC), Rutherford Appleton Laboratory</td>
</tr>
<tr>
<td>Chilton, Didcot, Oxon OX11 0QX, UK</td>
</tr>
<tr>
<td>Phone:</td>
</tr>
<tr>
<td>Fax:</td>
</tr>
<tr>
<td>Mobile:</td>
</tr>
<tr>
<td>E-mail: <a href="mailto:d.l.giaretta@rl.ac.uk">d.l.giaretta@rl.ac.uk</a></td>
</tr>
</tbody>
</table>
CONTENT

1 INTRODUCTION ...........................................................................................................7
  1.1 PURPOSE OF THIS DOCUMENT ............................................................................7
  1.2 HOW TO READ THIS DOCUMENT ......................................................................7
  1.3 APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS .............................8
  1.4 GLOSSARY .............................................................................................................8

2 THE CASPAR AUTHENTICITY MODEL ....................................................................10
  2.1 MANAGING AUTHENTICITY: A KEY CONCEPT ................................................10
  2.2 MANAGING AUTHENTICITY: CRITICAL ISSUES .............................................11
  2.3 TOOLS FOR MANAGING AUTHENTICITY .......................................................11
  2.4 USE CASES ..........................................................................................................12
  2.5 ELEMENTS OF THE CONCEPTUAL MODEL ....................................................14
  2.6 TECHNICAL SOLUTIONS .......................................................................................17
  2.7 PROCESS DESCRIPTION LANGUAGES ...............................................................17
  2.8 PROCESS EDITING TOOLS ....................................................................................17
  2.9 PROCESS EXECUTION TOOLS ...............................................................................17
  2.10 NEXT STEPS .........................................................................................................18
       2.10.1 Refining Conceptual Model ...........................................................................18
       2.10.2 Refining Use Cases .....................................................................................18
       2.10.3 Completion and Validation ..........................................................................19
       2.10.4 Preservability of the Evidence of Authenticity ...........................................19

3 DIGITAL RIGHTS MANAGEMENT IN CASPAR ....................................................20
  3.1 INTRODUCTION TO THE COPYRIGHT DOMAIN ...........................................20
       3.1.1 Outline of the Rights value chain .................................................................20
       3.1.2 Legal framework: rights beyond licenses and contracts ...............................20
  3.2 ANALYSIS OF DRM ELEMENTS APPLIED TO A PRESERVATION ENVIRONMENT 22
       3.2.1 Licensing ......................................................................................................22
       3.2.2 Digital Rights enforcement ..........................................................................22
       3.2.3 Enforcement technologies ...........................................................................22
       3.2.4 Rights Expression Languages .......................................................................23
  3.3 USE CASES ...........................................................................................................23
  3.4 ELEMENTS OF DRM CONCEPTUAL MODEL ..................................................26
       3.4.1 DRM Preservation .......................................................................................26
  3.5 COMPONENT INTERFACES ...................................................................................29
  3.6 TECHNOLOGIES AND STANDARDS FOR IMPLEMENTATION .........................30
       3.6.1 MPEG-21 REL .............................................................................................30
       3.6.2 MICC Watermarking Tool ...........................................................................31
  3.7 CASPAR RIGHTS ONTOLOGY .............................................................................32
       3.7.1 The domain of the CASPAR Rights Ontology .............................................32
  3.8 SAMPLE APPLICATION SCENARIOS ....................................................................34
       3.8.1 Request of a license on a digital object ..........................................................35
       3.8.2 Modification in the legal framework ..............................................................35
       3.8.3 Law introduces a Copyright Exception ..........................................................36
       3.8.4 Expiration date of Copyright changes in a country .....................................37
       3.8.5 Change in Public Domain ............................................................................38
       3.8.6 “Compulsory licenses” are introduced ..........................................................38
       3.8.7 Copyright levies .............................................................................................39
       3.8.8 Bilateral (or international) Agreements ..........................................................39
       3.8.9 Copyleft .........................................................................................................40

4 DATA ACCESS MANAGER AND SECURITY .......................................................41
  4.1 USE CASES ...........................................................................................................41
       4.1.1 Authentication Use Cases .............................................................................42
       4.1.2 User Management Use Cases ......................................................................43
5 IDENTIFIERS IN CASPAR

5.1 IDENTIFIER CLASSIFICATION ................................................................. 53
5.2 IDENTIFIER REQUIREMENTS ................................................................. 54
5.2.1 Resource Identifier Requirements .................................................. 54
5.2.2 Persistent Identifier Requirements ............................................... 54
5.2.3 Curation Persistent Identifier (CPID) Requirements ................... 55
5.3 IDENTIFICATION/RESOLUTION MODEL ............................................... 56
5.3.1 Identification domain definition .................................................... 56
5.3.2 URIs .................................................................................................. 56
5.3.3 Identification vs Localization ......................................................... 57
5.3.4 Model ............................................................................................... 57
5.4 CASPAR RESOURCE IDENTIFIER CANDIDATE .................................... 58
5.4.1 Resource versioning, logical and unique identifiers .................... 59
5.4.2 Use of Extensible Resource Identifiers (XRI) ............................ 59
5.5 CPID MODEL ......................................................................................... 60

6 INFORMATION DISCOVERY IN CASPAR ................................................ 61

6.1 REQUIREMENTS FOR FINDING AIDS ................................................. 61
6.1.1 Finding Aids and Descriptive Information in the OAIS Reference Model ..................................................................... 61
6.1.2 Additional Requirements ................................................................. 62
6.1.3 Discussion of requirements ............................................................. 63
6.2 A CONCEPTUAL MODEL OF FINDING AIDS .................................... 64
6.2.1 Functional View ............................................................................... 65
6.3 HIGH LEVEL INTERFACE SPECIFICATION ...................................... 66
6.4 LOGICAL VIEW ....................................................................................... 67
6.4.1 Finding Manager Interfaces Refinement ....................................... 67
6.4.2 Finding Manager main sequence diagrams .................................. 70
6.4.3 Finding Manager start up ............................................................... 72
6.4.4 Finding Manager store DescInfo objects ..................................... 73
6.4.5 Finding Manager discovery AIP ..................................................... 74
6.5 IMPLEMENTING A FINDING MANAGER ............................................ 75
6.5.1 Finding Manager based on SWKM (business and data layers) .... 76
6.5.2 Web Service interface for the Finding Manager (UI layer) .......... 77

7 CONCLUSIONS ....................................................................................... 80
1 INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT
The present deliverable reports on the first 18 months of activity of WP 2300, “OAIS-based access”. The objectives of this WP are “to specify and model knowledge components that will be based on the CASPAR Conceptual Model, and the CASPAR Framework architecture in order to implement a practical OAIS Access entity, using state of the art techniques and beyond, but always ensuring usability over the long term”.

WP 2300 is structured into the following tasks:
- Task 2304 - Unique identification, name resolution and resource discovery mechanism. The objective is to develop a uniform and robust system for persistent, unique and actionable identification of content pieces.
- Task 2303 - Authenticity management tools. The objective is to develop tools that ensure integrity and authenticity verification of contents and contextual information along the preservation process.
- Task 2302 – Digital Rights Management. The objective is to model DRM policies, leveraging emerging standards on Semantic description and inference.
- Task 2305 - Information access. The objective is to implement a component supporting the access to the information stored in an archive endowed with long-term preservation information.

Correspondingly, this report is divided into the following sections, Authenticity (Section 2):
- Section 2, on the authenticity model, contributed by University of Urbino;
- Section 3, on Digital Rights Management (DRM), contributed by Metaware;
- Section 4, on Access Management and Security, also contributed by Metaware; the topics of this Section were included in the general area of DRM, but for a better structuring of the deliverable (and of the project work) they are dealt with in a separate Section;
- Section 5, on Persistent Identifiers, contributed by (@Semantics and STFC); and
- Section 6, on Information Discovery, contributed by CNR. CNR also took over the responsibility of this Work Package and acted as an editor of the present document.

Finally, Section 7 concludes.

From a general point of view, this report should be understood, for each addressed component, as a refinement of the conceptual model, filling in the missing details where required, and as a design document leading to the first prototypes, expected to be delivered between month 24 and month 36 of the project. In some cases, implementation elements are presented, in some others implementation is only sketched because the work is at an earlier stage. The more technical documentation about design and implementation, as well as the implementation plan and its structuring iterations, are reported in the Architecture deliverable, to make the reading of this document easier and also to attain a uniform level of abstraction.

1.2 HOW TO READ THIS DOCUMENT
The Sections of this Deliverable may be read independently from each other and no particular order is required, since each one of them addresses a separate technical area of the project. The common conceptual background where each model is rooted is the CASPAR Conceptual Model, described in Deliverable D1201. The common technical background for the implementation of these components is the CASPAR Architecture, described in Deliverable D1301.
1.3 APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS

Applicable documents

(http://www.casparpreserves.eu/Members/metaware/ReferenceDocuments/caspar-description-of-work/at_download/file)

[D1201] CASPAR Conceptual Model – Phase I
(http://www.casparpreserves.eu/Members/cclrc/Deliverables/caspar-conceptual-model-phase-1-1/at_download/file)

[D1301] CASPAR Overall Component Architecture and Component Model
(http://www.casparpreserves.eu/Members/cclrc/Deliverables/caspar-overall-component-architecture-and-component-model-1/at_download/file)

Reference documents

[R1] CASPAR proposal, Sept 2005

1.4 GLOSSARY

[Ax] Applicable Document
[Rx] Reference Document
CASPAR Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval
DoW Description of Work
EC European Commission
EPM Executive Project Management
IPC IP Coordinator
IST Information Society Technologies
PACP Partner Administrative Contact Point
PO Project Officer
PPR Project Progress Report
PQE Project Quality Engineer
PTCP Partner Technical Contact Point
R&D Research and Development
SQE Stream Quality Engineer
ST Stream
TN Technical Note
WP Work Package
WPL Work Package Leaders

Designated Community An identified group of potential Consumers who should be able to understand a particular set of information. The Designated Community may be composed of multiple user communities. (OAIS definition)

Archival Information Package (AIP) An Information Package, consisting of the Content Information and the associated Preservation Description Information (PDI), which is preserved within an OAIS. (OAIS definition)
Content Information  The set of information that is the original target of preservation. It is an Information Object comprised of its Content Data Object and its Representation Information. An example of Content Information could be a single table of numbers representing, and understandable as, temperatures, but excluding the documentation that would explain its history and origin, how it relates to other observations, etc. (OAIS definition)

Knowledge Base  A set of information, incorporated by a person or system, that allows that person or system to understand received information. (OAIS definition)

Representation Information  The information that maps a Data Object into more meaningful concepts. An example is the ASCII definition that describes how a sequence of bits (i.e., a Data Object) is mapped into a symbol. (OAIS definition)
2 THE CASPAR AUTHENTICITY MODEL

2.1 MANAGING AUTHENTICITY: A KEY CONCEPT

Authenticity is a fundamental issue for the long-term preservation of digital objects: the relevance of authenticity as a preliminary and central requirement has been clearly investigated by many international projects, some focused on long-term preservation of authentic digital records both in the e-government environment, and in scientific and cultural domains (see for example www.interpares.org); some devoted to the identification of criteria and responsibilities for the development of reliable digital repositories (see for example the RLG-NARA recommendations Trustworthy Repositories Audit & Certification: Criteria and Checklist).

Defining and assessing authenticity are complex tasks and imply a number of both theoretical and operational/technical activities. These include a clear definition of roles involved, a coherent development of recommendations and policies for building trusted repositories, and a precise identification of each component of the custodial function. So, it is crucial to define the key conceptual elements that provide the foundation for such a complex framework. To this end the University of Urbino Team has provided an Authenticity Position Paper designed to share the knowledge about this issue. The paper tries to define how, and on what basis authenticity has to be managed in the digital preservation processes in order to ensure the trustworthiness of digital resources; and it also tries to define the conceptual basis of authenticity for the CASPAR project in terms of a common glossary. The glossary and the analysis of the key components of authenticity are based on the main results of international projects, and focused on the interconnections between these results and the OAIS conceptual model.

One of the founding concepts for the development of a theory on authenticity is that electronic resources can not be preserved as original, unchanged resources: we have only the ability to reproduce them. Unfortunately this quite obvious fact runs counter to the assumption that preserving authenticity implies retaining the identity and integrity of a digital object (what it purports to be and that is free from tampering or corruption). It’s a kind of paradox, where preservation requires changes, while authenticity requires fixity.

Authenticity cannot be recognised as given, once and for ever, within a digital environment. This point implies that a clear distinction should be made between the authenticity of a preserved resource (not necessarily the original one ingested in the repository), and the procedure of validating that resource: the latter is a part of a more general process aimed at assuring that an information object will be kept as an original one, i.e. reliable, trustworthy, and sound.

The authenticity of electronic resources is threatened whenever they are transferred across space (i.e. whenever they are sent to an addressee or between systems or applications), or time (i.e. either when they are in storage, or when the hardware or software used to store, process, or communicate them is updated or replaced).

Therefore, the preserver’s inference of the authenticity of electronic resources must be supported by evidence, provided in association with the resources through its documentation, by tracing the history of its various migration and treatments which have occurred over time. Evidence is also needed to prove that they have been maintained using technologies and administrative procedures that either guarantee their continuing identity and integrity or at least minimize risks of change from the time the resources were first set aside, to the point at which they are subsequently accessed.

In conclusion, authenticity is never limited to the resource itself, but is extended to the information/document/record system, and thus to the concept of reliability. Authenticity is also concerned with control over the information/document/record creation process and custody. The verification of the authenticity of a resource is related to the reliability of the system/resource, and this reliability should prove that it is fully documented with reference both to the creation process and to the chain of preservation.
2.2 MANAGING AUTHENTICITY: CRITICAL ISSUES

Integrity

The integrity of a resource refers to its wholeness. A resource has integrity when it is complete and uncorrupted in all its essential respects. The verification process should analyse and ascertain that they are consistent with the inevitable changes brought about by technological obsolescence.

The maintenance of the bit sequences is not always necessary, but the completeness of the ‘intellectual form’ is required, especially with respect to the original ability to convey meaning, e.g. maintenance of colours in a map, columns in a spreadsheet, etc. In other words, the physical integrity of a resource (i.e. the original bit stream) can be compromised, but the content structure and the essential components must remain the same.

Identity

A crucial point is that identity must be intended in a very wide meaning: the identity of a resource refers not only to its unique designation and/or identification. Identity refers to the whole of the characteristics of a resource that uniquely identify it and distinguish it from any other resource i.e. it refers not only to its internal conceptual structure but also to its general context (administrative, legal, documentary, technological, some could even add social).

2.3 TOOLS FOR MANAGING AUTHENTICITY

Requirements

Authenticity Management Tools have to monitor and manage protocols and procedures across the custody chain to deliver the benefits of authenticity into information system, from creation phase to preservation phase.

Authenticity cannot be evaluated by means of a Boolean flag telling as whether a document is authentic or not. Certainty about authenticity is a goal and cases where we can be sure are edge cases. We asymptotically approach authenticity. So we have to design all the mechanisms and tools keeping in mind that we could have alteration, corruption, lack of significant data and so on, and we need tools, mechanisms and weights to understand their relevance and their impact on authenticity. The consequence is that ensuring authenticity means providing a proper set of attributes related to content and context, and verifying/checking (possibly against metrics) the completeness or the alteration of this set.

Authenticity Management Tools have to identify mechanisms for ensuring the maintenance and verification of the authenticity in terms of identity and integrity of the digital objects. These tools have to provide content and contextual information relevant to authenticity, i.e. to the identity and integrity profile, along the whole preservation process.

The most critical issues are the right attribution of authorship, the identification of provenance in the life cycle of digital resources, the insurance of content integrity of the digital components and their relevant contextual relationships, and the provision of mechanisms to allow future users to verify the authenticity of the preserved objects or, at least, to provide the capability of evaluating their reliability in term of authenticity presumption.

So these requirements imply working on:

- Authorship attribution mechanisms and provenance control
- Content and contextual relationships
- Integrity control mechanisms
- Annotation process

Every profile has to be described and documented at every stage in the life cycle so as to have, any time it is needed, a sort of ‘Authenticity Card’ for any object in the repository.

Solutions
According to the above requirements, defining a strategy for managing authenticity means:

- identifying a set of attributes in order to catch relevant information for authenticity which can be collected along the life cycle of objects belonging to different domains. This means analysing the main and most promising metadata schemas and their basic components
- developing a conceptual model to describe the dynamic profile of authenticity i.e. to describe it as a process aimed at gathering, protecting and/or evaluating information mainly about identity and integrity

**Methodology**

Authenticity Team started mapping ISAD onto OAIS just to have a very general idea of some fundamental information elements which are to be preserved for ‘authenticity purposes’. This is assumed as a starting point to find some more elements by taking into account other resources (i.e. ISAAR, EAD, EAC, PREMIS, InterPARES). CIDOC CRM is going to be used and evaluated both as a suitable means of expressing concepts and as a resource giving us clues about relevant aspects needed for consideration, especially about dynamic aspects (temporal entities).

The work done so far has raised some issues:

- level of granularity. Authenticity fundamental requirements must be clearly identified in order to have neither overload nor exiguity of information
- variety of domains. Authenticity methodology and concepts are cross-domain but their deployment is strongly dependent on specific environment. For example:
  - the Reference Information for a book could be ISBN, very specific and not suitable for other typologies
  - the authorship concept is quite ‘easy’ for a book but what about the author of a movie?
- overlapping of concepts coming from different schemas. It’s not easy to decide whether an element has to be mapped onto either this or that OAIS conceptual element (e.g. whether the ISAD element “System of arrangement” belongs to either OAIS Provenance or OAIS Context). Anyway, the Authenticity Team recognizes that the its aim is to find a set of information elements and assign them to an OAIS category: it’s just a formal convention and so some uncertainties can be resolved

Of course, some critical points of the OAIS conceptual design are worthy of discussion: some categories and related semantics do not seem a perfect fit with the archival point of view. For example the OAIS categorization of PDI (Context, Provenance, Fixity, Reference) poses some semantic problems: Provenance (i.e. the history of Content Information) is part of the context because it refers to the context/environment of creation. To a careful analysis Reference (i.e. mechanisms to assign identifiers to Content Information) partially overlaps with the Context: the way things are denoted and/or classified changes during time, and so does the reference profile; but from an historical point of view this is a matter of context too, where ‘historical’ does not mean ‘long ago and therefore far away from us’, it refers to the history of the object. This history tells us a lot about the life of the object and it is something we cannot miss without losing control on authenticity. Clarifications of these distinctions could be a matter for later theoretical considerations about critical points in OAIS, with special reference to archival requirements. However these are not major issues since OAIS itself notes, in section 4.2.1.4.2, that “Provenance can be viewed as a special type of context information.”

### 2.4 USE CASES

**Use Cases Model**

The Authenticity Management Model includes, as a first approximation, the following use cases:

- **Gather/Create Identity Information**
- **Gather/Create Integrity Information**
- **Gather/Create Protocol and Procedure Information**
Gather/Create Identity Information

The *Gather/Create Identity Information* refers to the identification of information elements mainly related to

- provenance (with reference to archival history/chain of custody, origin or source, changes since the creation, …)
- context (scope and content, name of action or process, archival bonds, designation, extent, medium, taxonomic systems, reference systems, registration systems, …)
- conditions of access and use
- allied materials
- …

Gather/Create Integrity Information

The *Gather integrity information* refers to data integrity checks or validation/verification keys, and to procedures aimed at preventing, discovering, and correcting loss or corruption of records

- **Description:**
  
  Information is extracted from Fixity Manager to generate a set of data through which evaluate the integrity of the resource and hence its authenticity. The Data Preserver evaluates the quality of information and adds any missing data needed to fulfil authenticity requirements

- **Scenario:**
  
  1) PDS-Ingest receives an AIP with a storage request and sends a storage alert to Authenticity Management
  2) Authenticity Management gathers information from Fixity Manager and sends an authenticity report to PDS-Ingest, possibly requiring more qualified data
  3) When information is complete or cannot be refined anymore, Authenticity Management associates it to the resource and, in case, it sends an authenticity alert to PDS and to Administration entity

Gather/Create Protocol and Procedure Information

The *Gather/Create Protocol and Procedure Information* refers to definition and management of so called *Authenticity Protocols* that model a sequence of steps aimed at describing and evaluating the authenticity profile

The *Gather/Create protocol and procedure information* refers to protective strategies and/or solutions adopted in order to maintain authenticity or that affect authenticity in any way. The recursive design of information objects emphasizes the recursive nature of the problem of authenticity, so we need to manage authenticity of content/context information too, or at least define the policies for its control, for example by recording the responsibilities on creation and/or modification of content/context information

- **Description:**
Information is extracted from RepInfo, PDI, Migration, and Placement Managers to generate a set of data aimed at express the value of preservation chain or rather the overall quality of system in terms of authenticity management. The Data Preserver may add any useful information, including responsibilities on creation and/or modification of content/context information, and evaluation or description of authenticity of content/context information.

- **Scenario:**
  1) PDS-Ingest receives a storage request with an AIP and sends a storage alert to Authenticity Management
  2) Authenticity Management gathers information from RepInfo Manager, PDI Manager, Migration Manager and Placement Manager, and sends an authenticity report to PDS-Ingest, possibly requiring more qualified data
  3) When information is complete or anyway it can’t be refined anymore, Authenticity Management associates it to the resource and, in case, it sends an authenticity alert to PDS and to Administration entity

2.5 **ELEMENTS OF THE CONCEPTUAL MODEL**

**Authenticity Protocol (AP)**

The protection of authenticity and its assessment is a process. In order to manage this process, we need to define the procedures to be followed to assess the authenticity of specific type of objects

We call one of these procedures an **Authenticity Protocol** (abbreviated as AP). An AP is a set of interrelated steps, each of one is called **Authenticity Step** (abbreviated as AS). Every AS models a part of an AP that can be executed independently as a whole, and constitutes a significant phase of the AP from the authenticity assessment point of view. The relationships amongst the steps of an AP establish the order in which the steps must be executed in the context of an execution of the protocol. To model these relationships, we can use any workflow model. We do not enter into the details of this modelling here, and simply denote as Workflow the set of required relationships. The model introduced so far can be expressed in UML notation as follows:

![Figure 1 Authenticity Model](image)

**Authenticity Step (AS)**

An AS is performed by an **Actor**, who can act either in an automatic (hardware, software) or in a manual (person, organization) way:
Moreover, an AS is defined (possibly) following OAIS, we distinguish ASs based on the kind of Preservation Description Information required to carry out the AS. Consequently, we have 4 types of steps:

- Reference Step
- Provenance Step
- Fixity Step
- Context Step

Since an AS involves a decision, it is expected that it contains at least information about:

- the criteria that must be satisfied in taking the decision
- good practices or methodologies that must be followed
- the actors who are entitled to take the decision.

Moreover, an AS is defined (possibly) following Recommendations and is disseminated as established by a Dissemination Policy.
Authenticity Protocol Execution (APE)

APs are executed on objects belonging to a specific typology, in the context of Authenticity Execution Sessions. The execution of an AP is modelled as an Authenticity Protocol Execution (APE for short). An APE is related to an AP via the IsExecutionOf association and consists of a number of execution steps (Authenticity Step Executions, ASEs for short). Every ASE, in turn, is related to the AS via an association analogous to the IsExecutionOf association, and contains the information about the execution, including:

- the actor who did the execution
- the information which was used
- the time, place, and context of execution
- possibly the outcome of the execution. Not every step necessarily implies a decision, some steps simply imply collecting information related to a specific aspect of the object, e.g. title, extent, dates, and we are only interested into declaring the step has been done, without any form of evaluation. From a modelling point of view, we could classify steps as decisional (and the outcome is the decision) and non-decisional ones (having a different kind of outcome as an attribute, e.g. “step done” or “step not completed for such and such reason”).

The Authenticity Protocol Report will include also the information on which the execution/evaluation of Steps was based

Authenticity Step Execution (ASE)

Different types of ASEs will have different structures. Additionally, an ASE may contain a dissemination action.

Moreover, we are dealing with preservation and so we also want our model to be able to cope with the evolution of both APs and their executions over time. The evolution of an AP may concern the addition, removal or modification of one of the steps making up the AP. In any case, both the old and the new step are retained, for documentation purposes. When an AS of an AP is changed, all the executions of the AP which include an ASE related to the changed step, must be revised and possibly a new execution is required for the new (modified) step. Also in this case, the old and the new ASEs must be retained

Overall Authenticity Model

The overall model is designed as shown in the diagram below:
2.6 TECHNICAL SOLUTIONS

Authenticity as a Process

According to the above, the Authenticity Management component deals with Authenticity Protocols which are processes defined for specific types of objects in order to guarantee their identity and integrity. In this perspective, Create/Manage Authenticity Protocol and Execute Authenticity Protocol are the main features provided by the Authenticity Management component. And for the same reason the core of the Authenticity Management component is a Process Editor.

From a technical point of view BPEL could be a formalism to express (and execute) the Authenticity Protocol; JBOSS should have a BPEL engine and would therefore be a candidate for the CASPAR Framework (Planets also is adopting JBOSS, so it would be a strategic choice; an alternative is Eclipse that has a BPEL plug-in). The current state of the art for “Process Description Languages”, “Process Editing Tools”, and “Process Execution Tools” has been very briefly summarised by Luigi Briguglio in the CASPAR Authenticity Management: Model Refinement document and reported here.

2.7 PROCESS DESCRIPTION LANGUAGES

Currently there are two standard process description languages:

- XPDL defined from WFMC
- BPEL defined from OASIS. It is the process description language specification more supported by open source community and commercial tools

2.8 PROCESS EDITING TOOLS

- XPDL Editors:
  Enhydra JaWE (Java Workflow Editor) - [http://www.enhydra.org/workflow/jawe/index.html](http://www.enhydra.org/workflow/jawe/index.html)
- BPEL Editors:

2.9 PROCESS EXECUTION TOOLS
2.10 NEXT STEPS

The next steps in the development of the Caspar Authenticity Model will need an integrated action of refining concepts, integrating requirements, performing testbeds to evaluate models, and integrating with other work in the project.

2.10.1 Refining Conceptual Model

A general refinement action is needed for

- use cases
- domain classes of the Authenticity Management component
- Authenticity Manager Interfaces.

Authenticity protocol steps could have an outcome related to different dimensions (i.e. reliability, accuracy, identity, integrity etc), for example, a Fixity Step would typically have an impact on integrity. These interrelationships will be investigated.

2.10.2 Refining Use Cases

The use cases could be better designed by expanding the Gather/Create Protocol&Procedure Information and creating wrappers around the Gather/Create Identity/Integrity Information. So this could be a new design for the Use Cases of the Authenticity Component:

- Manage Authenticity Protocol (i.e. create, update, delete, and browse an Authenticity Protocol). It includes the use case Manage Authenticity Step
- List/Search Authenticity Protocols
- Execute Authenticity Protocol. It includes the use case Execute Authenticity Step
- Import/Export Authenticity Protocol
2.10.3 Completion and Validation

The development of the Authenticity Management Tool requires further work on attributes and features of domain classes to be defined. This will require us to:

- complete the definition of the information elements’ set, together with the related semantics, analysing different sources (ISAD, ISAAR, EAD, EAC, PREMIS, InterPARES, CIDOC CRM).
- continue the evaluation of watermarking techniques, and any other techniques (from checksum to digital signature) in relation to potential applicability and benefits for authenticity purposes, always keeping in mind that these mechanisms can change in time and must be documented.
- continue to cooperate with Engineering to be supported in selecting technologies and languages.
- validate the conceptual model, involving in particular the University of Glasgow and INA.
- involve testbed partners to test and verify the Authenticity Model and consequently refine it.
- investigate how best to integrate authenticity into archive Ingest components. Authenticity Protocols, as defined in the model proposed above, is strongly related to the Submission Agreement defined between a Producer and an OAIS. Working on Authenticity means capturing and possibly evaluating a specific set of information, so a model of the Submission Agreement should include asking the Producer for specific information elements at the timed of Ingestion.
- integrate with Finding Aids: the Authenticity Component is related to the Finding Aids Component, and in general with Dissemination, because Users may want to know about authenticity of the DIP provided, and so a template must be defined for delivering the authenticity information to the Users.

2.10.4 Preservability of the Evidence of Authenticity

The evidence about authenticity will be captured in our implementations using a variety of techniques e.g. CIDOC-CRM, BPEL. However we recognise that the encodings used will change according to the views of system users, and will certainly change over time and. We must therefore ensure that, as with any other Data Object, the appropriate Representation Information is captured and made available so that, in future, these pieces of evidence can be understood and used. Of course the evidence may itself be migrated to more currently usable forms, as might any other digital objects, but authenticity evidence itself may perhaps require special consideration in terms of being unchanged over time; further consideration will be given to this recursion.
3 DIGITAL RIGHTS MANAGEMENT IN CASPAR

In this chapter the digital rights application domain is outlined, including background information which introduces more preservation-specific issues related to DRM. Then the DRM component is described, including use cases, conceptual model and scenarios. The Rights Ontology is finally introduced, which is used to model and preserve some aspects of Copyright related to the archived items.

3.1 INTRODUCTION TO THE COPYRIGHT DOMAIN

3.1.1 Outline of the Rights value chain

The owner of copyright of a work is generally, at least in the first instance, the person who created the work, that is to say, the author of the work. Copyright comprises a set of rights that are bestowed by law on the owner of copyright, and are frequently described as exclusive rights to authorize others to use the protected work. The original authors of works protected by copyright also have moral rights, in addition to their exclusive rights of an economic character.

Related rights (i.e. the rights of performers, producers and broadcasters) can also be considered as economic rights limited to their contributions.

From this initial situation, copyright, with the exception of moral rights, may be assigned or licensed. Assignment means that the owner of the copyright transfers it to another person or entity, which becomes the owner of the copyright. Licensing means that the owner of the copyright remains the owner but authorizes someone else to exercise all or some of his rights, usually subject to some limitations.

This also applies to economic rights. Moral rights can be neither licensed nor transferred. Licensing achieves almost the same practical effect than assignment.

Each right defines and regulates a set of actions. Most copyright laws define the actions in relation to a work which cannot be performed by persons other than the copyright owner without the authorization of the copyright owner.

End-users do not hold any right. To exercise the right, i.e. to realise actions that are covered by copyright they should acquire a proper permission through a license. Moreover, the end-users might be subject to special conditions under which they have acquired the permission to use a work, and once they have the permission, it might also be constrained by some limitations.

We note that Copyright laws may give to the end-users some special permissions that grant them the possibility to perform some actions otherwise forbidden by copyright, and still without the authorization of the right holders. These exceptions to copyright (i.e. the User Rights) should be considered as end-user privileges and not rights. However they are here modelled as rights in order to build a more homogeneous model.

3.1.2 Legal framework: rights beyond licenses and contracts

As explained before, rights can be exercised also by persons that are not the copyright holders. In fact, permission to exercise the rights can be granted either by Law or by the rights holders. In fact, Rights holders may grant rights through licenses or they may even transfer some of the rights (i.e. “economic rights” and “related rights”) through contracts, while other special rights, or Copyright Exceptions, are granted to the end users by the legal framework. An example is Fair Use.

For this reason it is useful to give a complete picture of the legal framework that applies to the specific case, including spatial and temporal limitation of copyright, objects not governed by copyright (also called public domain), and rights granted to all users by Law. All this information should be registered
in CASPAR to extend the information that is otherwise only formalised within the licenses, and it will be contained in an object called Legal Framework Context, which is described in the next sections.

Both the Licenses and the Legal Framework Context objects will then have a semantic description that relies on a rights ontology.

![Diagram](image)

**Figure 5: Outline of the rights value chain**
3.2 ANALYSIS OF DRM ELEMENTS APPLIED TO A PRESERVATION ENVIRONMENT

Some preliminary considerations about DRM elements and digital preservation are given here to introduce the model for digital rights management within CASPAR, which is described later.

3.2.1 Licensing

The digital material that is submitted and ingested into the OAIS may be subject of some access restrictions. Thus, a license accompanies it during the distribution. An OAIS archive may need to support distribution along with licenses.

In the context of long term preservation, this is still true. This means that licensing information must be provided during submission of the SIPs. The semantics of the license must be preserved, but still considering that there will be changes in the legal framework that may affect the meaning of the license.

Moreover, the specification formalism of the license will not necessarily remain the same.

3.2.2 Digital Rights enforcement

An OAIS archive may wish to support DRM functionalities, and in particular, rights enforcement. This is applied at dissemination time, but has been specified at submission time within the Submission Agreement. However, technology evolves, and so do digital rights enforcement solutions. It is thus not adequate that the Submission Agreement template contains a technology-dependent specification about enforcement. It rather should contain a general and abstract description of the control and protection, for instance by means of the Producer’s expectations about functional aspects and security level.

3.2.3 Enforcement technologies

There are several technological measures that help in enforcing rights protection. The most used up to now are cryptography and watermarking.

Cryptography will perhaps mainly be applied to DIPs and SIPs, rarely to AIPs, while watermarking may be applied also to AIPs.

Watermarking is imperceptibly embedded information that is inserted directly into the content media. For this reason, it is necessary to keep track of the existence of the watermark within the Representation Information. Even if it is not necessary for the interpretation of the Data Object, it influences it. In fact, the migration of the Data Object into a different format requires that there is all the necessary information to deal with the watermark, i.e. to remove it if necessary, and to apply it also to the new Data Object.

Cryptography as securing measure

Cryptography may be applied to the Data Objects and to entire Information Packages. All kind of packages may be encrypted for securing their content during transmission. Thus, mainly SIPs and DIPs are going to be encrypted. However, also AIPs may be scrambled, i.e. in distributed and federated archives where transmission of AIPs over the network is required.

On the other side, only DIPs are going to contain even encrypted Data Objects, which are finally the rights protected objects of dissemination.

Watermarking as integrity and authenticity guarantee

There are several types of watermarks with different application scopes. In the context of digital preservation, watermarks can be used to support content authenticity. The market content is imperceptibly inserted into the content data object. Thus, it can be used as a technology to provide evidence of authenticity at data-level, i.e. considering that watermarks:

- are like tattoos, which can not be removed,
Two current standards are MPEG-21 REL and ODRL. These formalisms are called Rights Expression Languages (REL), and the developed to create machine-understandable licenses and to allow some enforcement measures that are under certain conditions, such as for a pre-view of a rendered object, such as an image, during the Consumer’s query and retrieval process.

3.2.4 Rights Expression Languages

In order to support some automatism in Rights Management, some specification formalisms have been developed to create machine-understandable licenses and to allow some enforcement measures that are driven by the license terms. These formalisms are called Rights Expression Languages (REL), and the two current standards are MPEG-21 REL and ODRL.

If a digital content needs to be archived along with its related license offer(s), this may be realized in several ways. The most naïve solution would be to use a traditional human-language license that accompanies the digital content. Another possibility would be to store the license terms in a specific REL language. An alternative could be to rely on a licensing concepts ontology to express the license.

In all cases, the license would be preserved by treating it as an instance of Content Information, where the Representation Information provides the way to interpret the license Data Object. In fact, the language used to express the license, which is a REL, an ontology or human language, evolves over time.

3.3 USE CASES

In the following section, the most important use cases that involve the DRM module are presented. They represent the high-level operations of the users, both internally and externally to the archive, and they give an overview of the requirements to be fulfilled by the DRM component.

The actors involved in the use cases are:

- License manager (archive external staff member) - The license manager has the ability to create license offers, assisted by a proper GUI. He/she is responsible for managing the license offers templates and the license offer documents. He/she is external to the OAIS archive, specifically his/her role would map to the “OAIS Management” entity.
• Ingest staff member (archive internal staff member) - The Ingest staff members are responsible, among others, for generating the Archival Information Packages (AIPs) and for coordinating updates to the Archival Storage and Data Management. They are involved in the operation of binding the content of AIPs to one or more license offer documents.

• DRM manager (archive internal staff member) - The DRM manager represents several experts, who have the ability to manage and administer all aspects of the DRM component. They belong to the OAIS internal archive staff; for instance they are responsible for updating the semantics of the license offers and license instances, but without updating the licenses themselves.

• Consumer - This actor includes the buyers of the licenses, namely the License Principals.

• Rights Holder - This actor represents the Rights-holders, namely the Content Creators and the Content Providers.

• Packaging module and Access module - These actors represent functional entities of the OAIS archive, which interact with the DRM module.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manage License Offers</td>
<td>Each content provider should be able to associate digital rights to one or more content objects. Only after this step the content can be available for distribution. Content Provider should be able to change or extend permitted uses for digital content items.</td>
</tr>
<tr>
<td>2</td>
<td>Instantiate License Instances</td>
<td>If a user wants to access a content object, a check on its linked license offers verifies if a license from the rights holder can be granted and how. The user has to follow all steps defined within the license offer in order to finalize the transaction and obtain the license for the content.</td>
</tr>
<tr>
<td>3</td>
<td>Invalidate License Instances</td>
<td>The Digital Right Manager might have the possibility to invalidate an instantiated license in case for instance a user has not followed the steps defined in the license offer or if he uses the content in a not authorized way.</td>
</tr>
<tr>
<td>4</td>
<td>Add Right Enforcer</td>
<td>The Digital Right Manager might have the possibility to add and integrate a new rights enforcing tool.</td>
</tr>
<tr>
<td>5</td>
<td>Apply Rights Enforcer</td>
<td>Once a transaction has been settled to access a specific content, the user can access a copy of it. The digital content should be protected through an enforcing algorithm (e.g. encryption, digital signature, watermarking, etc…).</td>
</tr>
<tr>
<td>6</td>
<td>Verify License Instances</td>
<td>Once a specific content has been requested the Data Access Manager and Security (DAMS) calls the DRM to verify if the resource is governed by a license and if the requester has a valid license for it. If everything is ok the user can go on and access a protected copy of the content.</td>
</tr>
<tr>
<td>7</td>
<td>Manage License Documents</td>
<td>The Digital Right Manager might have the possibility to update license offers according to modifications related to license parameters and/or values.</td>
</tr>
<tr>
<td>8</td>
<td>Manage License Language</td>
<td>The Digital Right Manager might have the possibility to export/import license offers and instances to/from a format which is compliant with the current ontology. This allows the re-definition of licenses in a different language.</td>
</tr>
</tbody>
</table>
The Digital Right Manager might have the possibility to update license offers rights and conditions according to modifications related to the legal framework. From the laws/regulations which has been affected by the modifications it is possible to retrieve the affected rights and therefore to retrieve the license offers and instances to be updated.

Figure 6: Use Cases Diagram
3.4 ELEMENTS OF DRM CONCEPTUAL MODEL

The role of DRM module inside the CASPAR Architecture is basically that of defining, distributing, enforcing, verifying and preserving digital rights on content and services.

In order to provide IPR (Intellectual Property Rights) protection, the following functionality is required:

- Digital rights definition and distribution: the DRM Layer allows defining and issuing rights. Licenses are defined in term of which parties are allowed to use a resource, which rights they have and the terms and conditions that have to be fulfilled in order to be able to use those rights.

- Digital rights enforcement: there is obviously no utility to define rights if they aren't enforced in any way. Moreover once outside the CASPAR environment, the digital content must still be protected and its usage be traced. This kind of protection is implemented into the DRM module as a Right Enforcing Service, which allows the integration of a content encryption service and provides the possibility to embed a reference to the license acquired in the form of a watermark or a digital signature.

- Digital rights verification: the DRM Layer cooperates with the Data Access Manager and Security (DAMS) Module to block every user operation that can violate rights defined using the DRM tools.

- Digital right preservation: a module for DRM Management and Preservation allows maintenance on the other modules in order to align the DRM functionalities with the evolutions in technology and in the semantic aspects of rights, e.g. by registering new enforcers and RELs, and updating rights license syntax and semantics.

In a more formal way we can describe the model by its base entities:

- **License**: this represents the basic structure of a license and indicates which parties (AbstractUser) are allowed to use a resource (AbstractResource), which rights (Right) they have and the terms and conditions (Constraint) that have to be fulfilled in order to be able to use those rights.

- **LicenseOffer**: a document produced by the Right Holder which specifies the resources which are governed by a license, the terms of the license, the issuer of the license and under which conditions (Condition) a user can obtain it.

- **LicenseInstance**: a document that instantiates a license offer document by specifying all missing parameters, namely the date of issue of the license and the user (principal) to whom the license has been delivered.

- **REL**: this class models the language used to define rights and licenses (e.g. ODRL, MPEG-21, etc…).

- **RightsEnforcer**: this class models the enforcing mechanism used to protect digital rights (e.g. encryption, digital signature, watermarking, etc…).
3.4.1 DRM Preservation

Preservation is one of the most important issues to be addressed. As noted above, the license would be preserved by treating it as an instance of Content Information, where the Representation Information provides the way to interpret the license Data Object. Initially this Representation Information will not support automation, as discussed in the CASPAR Conceptual Model [D1201], but we hope some progress to support automation will be made. In addition, the language used to express the license, which is a REL, an ontology or human language, evolves over time.

A specific DRM Management and Preservation module has been therefore introduced in order to do maintenance on the other modules and to align the DRM functionalities with the evolutions in technology and in the semantic aspects of rights, e.g. by registering new IPM tools and updating license offers as a consequence of changes in technology and in the legal framework. This module should allow one to:

- Modify license offers parameters and values.
- Define licenses in a different rights expression language.
- Integration of new rights enforcing tools.
- Handle possible modifications in the legal framework.

Modify license offers parameters and values. An updateLicenseOffer service is provided to allow the Digital Right Manager to modify license offers documents according to modifications related to license parameters and/or values.

Define licenses in a different rights expression language. CASPAR provides an extensible design for a generic Rights Definition Framework, supporting several Rights Expression Languages (RELs). This is necessary in order to fulfil the requirement of being preservable and reusable in the future, when
rights expression languages might improve and/or change. If this is the case an exportLicenses service is provided to translate all license offers and instances in a suitable format, e.g. expressed in terms of the adopted ontology and vice-versa the importLicenses service can transform license offers and instances back in the new REL language.

Integration of new rights enforcing tools. CASPAR provides an extensible design for a generic Right Enforcing Framework, supporting several implementations. This is necessary because each institution may already have acquired and be already using their own technologies and in order to fulfill the requirement of being preservable and reusable in the future, when rights enforcing technologies might improve and/or change. If this is the case a new right enforcing tool can be added and integrated in the CASPAR system through the addEnforcer service.

Handle possible modifications in the legal framework. The legal framework is the set of laws and regulations which refer to digital rights. This concept has been modelled in terms of Code, Law, Article and Paragraph, which can refer to specific economic rights, related rights, moral rights or user rights. Changes in the legal framework can occur, so CASPAR system provides the service to handle the consequences of such changes. In particular, the rights that are granted through licenses are maintained aligned and consistent with the current laws and regulations, and also the user rights (“Copyright Exceptions”) which are granted directly by Law must be maintained updated. All this Legal Framework elements should be registered in CASPAR to extend the information that is otherwise only formalised within the licenses, and it will be contained in an object called Legal Framework Context.

The Legal Framework Context of a certain country will be gathered at dissemination time for those archive holdings that are governed by copyright. To that purpose a set of Legal Framework Context objects are constructed and preserved.

The LFC objects consist conceptually in the set of laws, regulations and agreements that are applied in a certain country.

![Figure 8 Legal Framework Model](image)

Advantages of a formal representation of the legal framework are:

- to provide supplementary/complementary information about permissions allowed by Law, rather than by License
- automatic evaluation of permissions taking legal framework into account
- to facilitate the preservation of (user) rights, being dependent from the legal framework

A set of scenarios is given later, which depicts specific cases of changes in the legal framework, and how the DRM component is involved to handle them.
3.5 COMPONENT INTERFACES

The DRM component implements the following interfaces:

- RightsDefinitionManager
- RightsDistributionManager
- RightsEnforcingManager
- RightsVerificationManager
- DRMReservationManager

```xml
<interface>
   RightsDefinitionManager
</interface>

```
3.6 TECHNOLOGIES AND STANDARDS FOR IMPLEMENTATION

3.6.1 MPEG-21 REL

The Mpeg-21 REL (Rights Expression Language) provides a complete model for declaring rights, and enables interoperability at import and export phases. Being based on XrML makes this format the most common in situations where digital rights enforcement has to be implemented in a strict and granular manner.

MPEG REL states that a license must specify at least:

- One or more Grants. A grant gives access to subject the right to use a content item by respecting some conditions. Several Grants can be contained into an Item.

- The license issuer. Typically, the issuer digitally signs the license, signifying that the issuer does indeed certify the grants that the license contains. In addition, the issuer may provide additional information about the issuance of the license.

- Miscellaneous additional administrative information.

The following elements are most commonly found in a REL license:

- LicenseID. Specified by an attribute of the license element, this identifier can be used to store and retrieve the license.

- Title. A description of the license in human readable form.

- Grant. Describes the rights identified by the license. Each Grant defines the right to use a resource to a party subject to certain conditions. The license has no meaning if it doesn't contain a Grant (or a GrantGroup).

- GrantGroup. Grouping the Grants into a GrantGroup allows using them as a single unit. This is useful for example as they can be billed together.

- Issuer. Information about who is issuing the license is specified using one or more issuer elements.

Other elements provide commodities, to simplify the license representation in XML format.

- Inventory. An Inventory contains declarations of terms used multiple times into the license, such as subjects. Once declared into the inventory, these can be directly referenced using the provided identifiers.
• Other info. Provides extensibility mechanism for the license, as any other relevant element not
defined into the specification may be inserted. Extensive usage of this field may pose
problems on interoperability.

Another relevant element in license declaration is that of defining conditions to be satisfied before a
right could be granted. Conditions can include, for example, a time interval within which a right can be
exercised, a limit to the number of times a right can be exercised, a fee to be paid before obtaining the
right or if another prerequisite right is required for exercising the right.

A single condition can itself be expressed in several ways. For instance the fee required to access
content can be:

• flat fee: one-time fee due when a principal exercises a right
• per use: fee due each time a principal exercises a right
• per interval: fee due for exercising the right for a given time frame
• metered: fee is due only for the time frame the right is exercised

3.6.2 MICC Watermarking Tool

Media Integration and Communication Center (MICC) watermarking is a particular watermarking
implementation which provides all the basic functionalities offered by this technology.

The watermark embedding service allows the digital rights owner or the authorized entity to perform
watermark insertion. Information needed during invocation of this service is:

• The digital content to be protected (e.g. the digital image).
• IPR/license information.
• A password, which is a private key. This key is composed by two fields:
  o First field: maximum 16 characters.
  o Second field: maximum 8 digits.

The watermarking embedding service returns the watermarked content (image) which appears
perceptually similar to the original one so its quality and its worth is preserved.

The watermark detection service takes as input:

• The digital content to be checked.
• A password, which is the same private key that has been used during the watermark
  embedding phase.

The watermark detection service returns the watermark string that contains IPR/license information.
This information permits, for instance, to make an assertion about the legal owner or the authorized
user of a specific digital asset. It can also be used successively to track the digital object distribution
and to trace the End User. If an illegal use of this good has been found, the End User may be
prosecuted for disseminating the content bought for one usage and used for another one.
3.7 CASPAR RIGHTS ONTOLOGY

A Rights Ontology is being developed in order to formalise copyright information that are related to archived objects. In particular, there are some rights metadata that are bound to the particular copyrighted content items (e.g. rights holders, publication year and country, existing rights on this item, related works, associated license offers), and there is some copyright information that is independent from single content items, i.e. the representation of the different Legal Frameworks, which must be taken into consideration depending on the country where the content is disseminated.

The concepts of the Ontology are based on the WIPO (World Intellectual Property Organisation) recommendations, which try to define a common worldwide legal framework. Using this general framework helps building a general copyright ontology, which can then be applied for any particular local copyright system.

Furthermore, it is important to note that the Rights Ontology has been harmonized as much as possible with the other ontologies that model general or even specific aspects of the items archived within the CASPAR framework. This is in order to avoid introducing semantic inconsistencies or redundancies. Furthermore, the rights information related to an archived item are not all clearly separable from other metadata. For instance Provenance and Context might need to be represented together, and they contribute in providing both Authenticity and Rights documentation. So, to integrate the diverse aspects of documentation, a common CASPAR Ontology is used, which provides a single consistent formal language. The Rights Ontology will then deal only with the description of the concepts and relationships that are related to the rights.

3.7.1 The domain of the CASPAR Rights Ontology

The overall CASPAR rights ontology will consist in: a core copyright ontology, a legal framework ontology, and a licensing concepts ontology.

The core copyright ontology will model the concepts that occur in the copyright domain, i.e. the entities and the relationships between them. Among the entities there are the human actors, the specific rights that govern very precise actions, the creative works being modelled as legal objects, actions that involve, use or bring into existence new creative works, as well as the relevant properties that correlate these classes. This part will be used to document the status of rights pertaining to single archived items.

The legal framework ontology will model the concepts that occur in the legal domain, such as Regulation, Agreement, Contract, License, Legal Code, Article, Law, Paragraph, etc. and these will be related to each other and to some entities of the core copyright ontology. This part will be used to document the relationships between the rights and the source texts.

The licensing concepts ontology will model the concepts that occur in the licensing domain, such as permissions, conditions, constraints, permission issuer and permission principal, and related also to the other entities of the overall Rights Ontology. It represents also a single canonical backbone language to be used for the integration of different existing RELs into the CASPAR DRM tool; it contains basic entities and relationships common across the diverse rights metadata vocabularies and the mapping with some RELs. This part will be used to document the license offers that are attached to the archived items.

Even though these three sub-domains will be used separately and play a different role within the DRM module, they are semantically tied, and their entities need to be consistently related to each other. They all provide, separately as well as totally, a lexical and taxonomical framework that will play the role of Representation Information.
Figure 9: CASPAR Rights Ontology framework

The following diagram outlines how some entities of the Rights Ontology are modelled within the DRM component.
3.8 SAMPLE APPLICATION SCENARIONS

In the following some scenarios are aimed at clarifying how the DRM component works. The first two scenarios are more high-level and highlight the interactions with other CASPAR components, while the latter represent more specific examples, i.e. copyright changes in the long term.

In fact, one of the main objectives of the DRM component within a digital preservation environment is to facilitate in "propagating" the effects that changes in the legal framework (law, international regulations and agreements between countries) have on the rights associated to archive holdings. In particular, the producer will associate license offers to the holdings at ingestion time. Then, if
necessary, they should be updated, so that at dissemination (distribution) time, the "consumer" can choose among a set of up-to-date license offers.

3.8.1 Request of a license on a digital object

Here is a sequence diagram describing the rights distribution, verification and enforcing process. This justifies the interaction among DRM and DAMS (AuthorizationManager).

The information flow for acquiring a digital right on a digital content item is the following:

1. A user selects a content object he has found searching in the CASPAR archive. If the content is protected by digital rights and the rights holder was willing to give a usage licence, the user is presented a list of license offers. He chooses one amongst them and a new user license is generated.

2. The user requests to access the content object by invoking a service. Before the service can be invoked, a security check has to be performed by the Data Access Manager and Security (DAMS) module. One of the tasks of this security check is to ask the DRM layer whether the content is protected by digital rights and, in case of a positive answer, if the user owns a valid license. If this verification is successful the DAMS grants access to the resource letting the service to be executed.

![Sequence Diagram](image)

Figure 11: Rights Distribution, Verification and Enforcement

3.8.2 Modification in the legal framework

Here is a sequence diagram describing the preservation of the license offers after a modification in the legal framework. This justifies the interaction of DRM with NotificationManager (Preservation Orchestration Manager) and with SWKMWebServices (Knowledge Manager).
3.8.3 Law introduces a Copyright Exception

This is an example of modification in the legal framework.

Description

A national law or international treaty introduces a new Copyright Exception, i.e. it grants to the end users the permissions to perform certain actions, as long as they comply with some precise constraints. An example is the definition of the “Fair Use” in the Statute Section 107 of the Copyright Act of 1976, which introduces the educational right, i.e. this Statute gives the authorization to use by reproduction in copies or phonorecords or by any other means specified in that section, for non-profit educational purposes.

The effect of this change should result in updating the Article within the Legal Framework description, and in updating the related Framework Context objects.

Sequence Diagram

Depending on the particular change in the Legal Framework, the above sequence diagram could have a variant, i.e. concluding with the update of a Legal Framework Context object, instead of the update of some license offers.
3.8.4 Expiration date of Copyright changes in a country

This is an example of modification in the legal framework.

Description

A national law or international treaty introduces a modification in the expiration of Copyright, e.g. extending it from 50 to 70 years after the death of the author.

The effect of this change should result in updating the Article within the Legal Framework description, and in updating the Framework Context objects of the involved countries.

The death dates of right holders are monitored by PDI Copyright data preservers, as well as possible events that influence the extensions of copyright, such as the prolongation of copyright extension by the author’s relatives.

At dissemination time the actual copyright expiration date is retrieved on the basis of the local legal framework context.

When the deadline is reached no licenses are needed anymore.

Sequence Diagram
3.8.5 Change in Public Domain

This is an example of modification in the legal framework.

Description

A national law or international treaty introduces or changes the concept of Public Domain. In particular, certain categories of creative works can be excluded from Copyright. Some concrete examples are: immaterial works, legal or administrative text from official bodies, statistics, phone books.

The effect of this change should result in updating the Article within the Legal Framework description, and in updating the related Framework Context objects.

Furthermore, at dissemination time, if the work is under Public Domain in the country of the end user, no license offer should be associated with the work.

Sequence Diagram

![Sequence Diagram](image)

Figure 15: Change in Public Domain

3.8.6 “Compulsory licenses” are introduced

This is an example of modification in the legal framework.

Description

In a compulsory license, a government forces the holder of copyright to grant use to the other parties. Usually, the holder does receive some royalties, either set by law or determined through some form of arbitration.

Compulsory licenses result from the operation of Law and not from the exercise of the exclusive right of the copyright owner. The remunerations resulting from compulsory licenses are usually collected by collective management organisations. These organisations license use of works and other subject matter that are protected by copyright whenever it is impractical for right owners to act individually.

For instance these compulsory licenses are required to obtain the right of public performance: music played or performed in discotheques, restaurants, and other public places. Another example could be the “orphan works”, namely copyrighted works where it is difficult or impossible to contact the rights holder.

A concrete example of the first compulsory license is described in:

http://www.copyright.gov/docs/regstat031104.html
3.8.7 Copyright levies

This is an example of modification in the legal framework.

**Description**

Some countries, e.g. Germany, impose copyright levies on VCRs, CD or DVD burners, and CD and DVDs, PCs and every other thing that is able to copy media content, be it music or television. With the current model of the DRM framework in CASPAR it is possible to handle a scenario where particular license offers would be offered to users of such countries, in order to take into account the remuneration they have already paid in form of copyright levies and to avoid a double taxation.

3.8.8 Bilateral (or international) Agreements

This is an example of modification in the legal framework.

**Description**

Bilateral (or international) agreements between some countries might for instance allow authors to claim copyright protection in a foreign country, despite in general the legal framework of the end user country is applied.

If the international agreement implies modifications in the Copyright Laws of the countries involved this scenario falls back into one of the other Preservation Scenarios.

Otherwise if such agreement applies only in certain cases and doesn't therefore imply a modification in the Copyright Laws, there are two ways to handle this issue:

- case 1: the LFC of the countries involved (in particular the "Agreement" section) is updated
- case 2: a new LFC associated to the agreement is created which substitutes the default LFC of a country if the conditions of the agreement are met.
3.8.9 **Copyleft**

This is an example of modification in the legal framework.

*Description*

Copyleft is a form of licensing and may be used to modify copyrights for works. Through a copyleft licensing scheme, every person who receives a copy of a work obtains the permission to reproduce, adapt or distribute the work as long as any resulting copies or adaptations are also bound by the same copyleft licensing scheme. A widely used and originating copyleft license is the GNU General Public License. Similar licenses are available through Creative Commons - called Share-alike.

Instead of allowing a work to fall completely into the public domain (where no copyright restrictions are imposed), copyleft allows an author to impose some but not all copyright restrictions.

This scenario represents a particular case of license offer creation (or update) by means of the rights holder.
4 DATA ACCESS MANAGER AND SECURITY

The first part of this chapter identifies the most relevant use cases involving user authentication and authorization. The analysis of these use cases produces inputs and requirements which serve as guidelines to build the Data Access Manager and Security (DAMS) component: from the conceptual model to the definition of the interfaces and the identification of standards and technologies to be adopted in the implementation phase. Finally some application scenarios are presented in order to give an example of how to use the services and functionalities provided by the component.

While we expect that most if not all archives will have their own equivalent of a DAMS system, and would be unwilling to change to the CASPAR approach, nevertheless we believe, in the interests of completeness, for our own use, and for a conceptual model onto which other systems can be mapped, that the DAMS component has an important role to play in CASPAR.

4.1 USE CASES

In the following section, the most important use cases related to user management, authentication and authorization are presented. They represent the high-level operations of the users, both internally and externally of the archive and give an overview of the requirements to be fulfilled by the DAMS component. Most of the requirements deal with how CASPAR actors interact with the framework services and resources.

The actors involved in the use cases are:

- User (CASPAR archive system user) - This actor represents any user of the CASPAR archive system, from Content Creators/Providers to Final Consumers.
- Administrator (Archive administrator) - This actor represents the archive administrator, who is responsible for user profiles, authorized communities and permissions management.
4.1.1 Authentication Use Cases

Authentication is the process that asserts that a subject is who she claims to be: this is proven by making sure that a set of credentials is presented by the subject each time she wants to prove her identity. Once authentication has been performed a user becomes authenticated.

Before making use of a service a user has to be registered and authenticated, i.e. his personal data and credentials must be stored and can be retrieved by the service.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Register</td>
<td>Through the Registration use-case a user provides some information on his profile and chooses the credentials which will be used to uniquely identify him.</td>
</tr>
<tr>
<td>2</td>
<td>Unregister</td>
<td>A user who is registered to a service can unregister from it. She will have to register again if she wants to continue to use it.</td>
</tr>
<tr>
<td>3</td>
<td>Login</td>
<td>A user provides his credentials to prove his identity before being allowed to use a service. A new working session begins with user login.</td>
</tr>
<tr>
<td>4</td>
<td>Logout Enforcer</td>
<td>A user terminates her working session. From this moment on the user is considered an anonymous user.</td>
</tr>
</tbody>
</table>
4.1.2 User Management Use Cases

The following requirements describe how users’ accounts are created and grouped into authorized communities. User accounts maintain user information about the identity of a user while authorized communities represent aggregation of users.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Modify User Account</td>
<td>The user is identified by a user name, that has to be unique, and a security token that proves her identity (AbstractCredentials). Other information can be relevant for maintaining a user database (email, full name, organization, credit information, etc…). Such information is stored along the user profile (AbstractUser).</td>
</tr>
<tr>
<td>2</td>
<td>Manage authorized communities</td>
<td>Authorized communities are defined by a name and a list of users or properties. The name must be unique. The property can restrict user membership only to users associated to a particular resource.</td>
</tr>
<tr>
<td>3</td>
<td>Update community members/properties</td>
<td>User management has to provide means to add users to a community (UserAuthorizedCommunity) and to define a community through one or more properties (PropertyAuthorizedCommunity).</td>
</tr>
</tbody>
</table>
4.1.3 Authorization Use Cases

Authorization is a part of the operating system that protects computer resources by only allowing those resources to be used by resource consumers that have been granted authority to use them.

The following requirements describe how permissions to exercise actions on resources are created and the process of creation of an access control policy.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manage permissions</td>
<td>Permissions are a literal values that represent one or more operations of the system. They are defined by a name and a list of actions. The name must be unique.</td>
</tr>
<tr>
<td>2</td>
<td>Bind/Unbind permissions to actions</td>
<td>Policy management has to provide means to associate/remove permissions to/from an action.</td>
</tr>
<tr>
<td>3</td>
<td>Manage rules</td>
<td>Rules are defined as triples composed of an authorized community, a resource and a list of permissions. The issuer of the rule is also specified.</td>
</tr>
<tr>
<td>4</td>
<td>Manage policies</td>
<td>Policies are defined by a set of rules which constitute an access control policy.</td>
</tr>
<tr>
<td>5</td>
<td>Add/Remove policy rules</td>
<td>Policy management has to provide means to add/remove rules to/from a policy.</td>
</tr>
</tbody>
</table>
Figure 21: Authorization Use Cases
4.2 ELEMENTS OF CONCEPTUAL MODEL

CASPAR access control model is mainly based on the Rule Role-based access control (RBAC) approach (see section 4.4.1). RBAC provides user authorization and access control in an elegant way through the definition, enforcing and evaluation of access control policies. For each resource, an access control policy can be declared within the security manager, binding users to permissions (rights to execute operations). The DAMS acts effectively both as a Policy Enforcement Point and a Policy Definition Point, as it lets administrator define policies and then assures the enforcement of these policies.

This model has been slightly modified and extended to encompass allowing the ability to personalize the concept of role and to preserve and re-use the system in the future. In this sense the concept of role, which is the key point of this model, has been modified into that of authorized community: in this interpretation an authorized community is just an aggregation of users. In the definition of an access policy it is possible to associate permissions to authorized communities. A user can access services and resources according to the permissions granted in the policies to the authorized community he belongs to.

The authorization model realised by the DAMS will have the following targets:

- the consumers are represented by system users (on behalf of which some code is executed)
- the resources are represented by OAIS Information Objects
- the actions are represented by service executions, i.e. by the methods of the components’ APIs

The policies defined and enforced by the DAMS will focus on authorizing users, and not on authorizing components within the system architecture. This is mainly because the final scope is to contribute with some preservation-specific capabilities, and the most significant issues in the long term preservation are raised due to evolution of rights of users with respect to information objects.

In particular, a requirement in CASPAR Access Authorization is the support for a flexible specification and execution of an authorization policy. For instance, the consumers could be represented by a group of future system users which have in common some property. By consequence, the conditions to authorize access could require human intervention due to their complexity and heterogeneity.

Authorization must be handled at two different levels: a static one that defines basic policies for accessing services and content, and a dynamic one that overrides the static policies if particular conditions are required (ex. a license is required for getting the content). Thus this functionality is linked to the DRM module.

When an actor tries to access a service or content the following procedure must be followed:

- The content or service is checked against the related security policy.
- A check is made to verify if the user has the right to perform the required operation according to the static permissions.
- When content is governed by copyright restrictions, a check is made if the user has a valid license to access/use the content.

In a more formal way we can describe the model by its base entities:

- **AbstractUser**: a person or external system that wants to access the CASPAR system. Each user belongs to one or more authorized communities, and these authorized communities define the way the user interacts with the system.

- **AuthorizedCommunity**: an authorized community can be define in a very general and open way: might have an extensional definition (**PropertyAuthorizedCommunity**), where membership is defined through one or more properties written in human language format or it might have an intentional definition (**UserAuthorizedCommunity**), where users belonging to
the community are listed with their identifiers. An example is a 'group' of 'registered users of the system'. But in general it could be a set of people that are not (yet) users of the system.

- **Permission**: is a literal value that represents one or more operations of the system. For example the "manage user" permission can include the "modify", "add" and "delete" operations for user accounts. Once this relation occurs we can say that a feature (or service) is protected by a given permission.

- **AbstractAction**: is a feature of the system. For example if every interaction with the system is done via a service invocation an action represents a service invocation. An action can have a target resource (i.e. it can be executed on a resource) or not.

- **AbstractResource**: represents the resource which is involved in the execution of an action.

- **Rule**: is a triple composed of an authorized community, a resource and one or more permissions. Rules are the access control engine because they identify what kind of actions (associated to permissions) users (aggregated in authorized communities) can perform on a resource.

- **Policy**: is a set of rules.

---

**Figure 22: DAMS Conceptual Model**

---
4.3 COMPONENT INTERFACES

In order to fulfill the requirements identified by the use cases analysis and to implement the conceptual model presented above, the DAMS component defines and implements the following interfaces:

- UserManager
- AuthenticationManager
- AuthorizationManager

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserManager</td>
<td>Allows management of users.</td>
</tr>
<tr>
<td>AuthenticationManager</td>
<td>Enables authentication process.</td>
</tr>
<tr>
<td>AuthorizationManager</td>
<td>Manages authorization of users.</td>
</tr>
</tbody>
</table>

```java
public interface UserManager {
    boolean isUserAuthorizedCommunity(String usersRepresentedByCommunity, AuthorizedCommunity authorizedCommunity);
    // Other methods...
}
```

```java
public interface AuthenticationManager {
    void authenticateUser(String username, String password);
    // Other methods...
}
```

```java
public interface AuthorizationManager {
    boolean isUserAuthorizedCommunity(String usersRepresentedByCommunity, AuthorizedCommunity authorizedCommunity);
    // Other methods...
}
```
4.4 TECHNOLOGIES AND STANDARDS FOR IMPLEMENTATION

4.4.1 RBAC

Several access control models have been developed during the years for modern applications and operating systems. One of the most relevant is Role Based Access Control. Role Based Access Control (RBAC) introduces the concept of user roles. The administrator defines a set of roles and assigns them to subjects. Several roles can be defined to map different interactions with the system. Resources and operations are then mapped to roles, so that users with that role can have access to them. Using this model the administrator can define access policies without having to set per-user permissions.

An important variant of RBAC is the Rule Based Access Control, which adds a set of rules to the access control policy evaluation. An example of such rule can be a predicate on the source IP address the operation is requested from. This approach is particularly suited when there is the need to model distributed system security policies.
4.4.2 XACML

OASIS XACML is an XML language that allows the declaration of very fine-grained access control policies, based on subject attributes, resource attributes and other conditions. It also allows policies to be shared by different hosts.

The key constructs used for defining policies using XACML are Policy and PolicySets.

- A PolicySet is a container that can hold other Policies or PolicySets.
- A Policy is constituted by a set of individual Rules or Policies.
- A Rule is a boolean expression over a Target which is a set of Subjects, Resources and Actions.
- An authorization decision is a request of executing an operation. When multiple Rules or Policies have to be evaluated for taking an authorization decision, a combining algorithm has to be adopted.
- A combining algorithm could be rule combining if arrives to an authorization decision by combining result of single rules evaluation, or policy combining if it evaluates the single policies and then combines these results. The available combining algorithms are: Deny overrides (if one rule or policy denies access then access is denied), Permit overrides (same as Deny, just replace Permit), First applicable (the first applicable policy or rule gives the authorization decision) and Only-one-applicable (just one of the Policies must be applicable).

As already mentioned, policies and rules are applicable to targets: rules can apply to a subject (user) if conditions on its identity, authorized community, or attributes are met. In the same way one could specify policies based on resource attributes. In similar fashion the operation to be executed by the subject on the resource, can be put as condition. Most commonly a Policy may just identify a set of users (authorized communities) that are allowed to perform a particular action on a particular resource.

4.5 SAMPLE APPLICATION SCENARIOS

Here are some sequence diagrams describing the access control process: policy definition, policy verification and preservation of the authentication information. This justifies the interaction of DAMS with RegistrationManager and NotificationManager (Preservation Orchestration Manager).
4.5.1 Creation of an access control policy

4.5.2 Authentication of a user
4.5.3 Modification in the definition of an authorized community

4.6 PRESERVATION OF DAMS INFORMATION

Further examination must be given to the extent to which the DAMS information can and should be preserved. Clearly one should capture Representation Information about the way in which the policies, roles etc are encoded, as with any other Data Object. Initially this Representation Information will almost certainly not support automation, but such support is obviously our eventual aim. The certificates or key stores may be somewhat more problematical in that one must take into account the possibility that the security encryptions for example may be “cracked” and so use of such a security mechanism, which may be of interest in terms of consideration of authenticity, needs further work.
5 IDENTIFIERS IN CASPAR

Identifiers play a number of important roles in any system; the two of particular interest here are (1) naming something and (2) locating something. The first of these allows one to, for example, find additional information about something one already has; the second allows one to obtain something (or a copy of a digital something) for which one has the identifier.

In this section we are mostly interested in the second role, sometimes referred to as an actionable identifier.

It also seems useful to distinguish:

- “normal” systems where the identifiers exist within the context of some system which is not expected to last over the long-term.
- distributed, asynchronous preservation systems where more requirements come into play.
- Preservation systems where new types of identifiers are needed which have some special requirements.
  - we focus mostly on persistent identifiers for Representation Information, but a number of other types of persistent identifiers are relevant.

Section 5.1 gathers requirements for identifiers. The following section discusses some general lessons to be learned from other identifier systems, then we present a fairly general model for identifiers. Section 5.3 describes the types of “ordinary” (short-term rather than long-term) identifiers we will use within the deployed components of CASPAR. Finally we describe what we call Curation Persistent Identifiers (CPIs), which are the most important persistent identifiers within the system, which is not itself a new “identifier system” which others must adopt, but rather a way of pooling risk between existing identifier systems.

5.1 IDENTIFIER CLASSIFICATION

In line with the ideas introduced in this section we present a classification of identifiers which we believe will be useful in what follows, and will be based on the OAIS Information Model.

![Figure 23 OAIS Information Model](image-url)

The relationship between these identifiers is not simple, as will be seen in the examples following the classification.

Figure 23 shows the OAIS Information Model, and for each of the components (except the "Bits") an identifier can be associated. A proposed taxonomy of identifiers which may be useful in a preservation...
system. The idea is to have a general Identifier which allows access to one or more Locators (see below). It is useful to distinguish some perhaps transient identifiers:

<table>
<thead>
<tr>
<th>OAIS object</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Object</td>
<td>Physical Object Identifier</td>
<td>An Identifier which provides the location of a physical object. The OAIS Physical Object is defined as: An object (such as a moon rock, bio-specimen, microscope slide) with physically observable properties that represent information that is considered suitable for being adequately documented for preservation, distribution, and independent usage.</td>
</tr>
<tr>
<td>Information Object</td>
<td>Information Object Identifier</td>
<td>Note that we will only be dealing with these as non-persistent identifiers in general.</td>
</tr>
<tr>
<td>Data Object</td>
<td>Data Object Identifier</td>
<td></td>
</tr>
<tr>
<td>Digital Object</td>
<td>Digital Object Identifier</td>
<td></td>
</tr>
<tr>
<td>Representation Information</td>
<td>Curation Persistent Identifier (CPID)</td>
<td>A special type of Persistent Identifier which is used with a preservation system to point to Representation Information.</td>
</tr>
</tbody>
</table>

As an example a book on a shelf needs a Physical Object Identifier e.g. the name of the book and where it may be located. The book may be a Data Object. If the book is regarded as needing no Representation Information (for its Designated Community) then it may also be regarded as an Information Object. If it is also a piece of Representation Information then it will need a CPID.

In addition we will need an identifier which will point to various resources such as CASPARD Key Components.

5.2 IDENTIFIER REQUIREMENTS

Within CASPARD we need identifiers to perform a number of functions.

5.2.1 General Resource Identifier Requirements

Identifiers are needed for a variety of resources such as Registries and other Key Components as described in the Conceptual Model [D1201] and Architecture [D1301]. These identifiers may not be persistent but are likely to have an appreciable lifetime as part of the information discovery process.

5.2.2 Persistent Identifier Requirements

The most fundamental requirement is the need for Persistent Identifiers – but for our purposes we do not need the most general purpose persistent identifiers capable of identifying any and all digital objects.

- A large amount of work has already been carried out on this general topic, some (not exhaustive) summaries of which may be found at:
5.2.3 Curation Persistent Identifier (CPID) Requirements

For CPIDs we can propose the following requirements:

- Actionable identifiers to Representation Information (RepInfo) i.e. pointers (probably encoded as character strings) able to be resolved and allow us to obtain a copy of the Representation Information from some type of Repository (in which case we refer to a Registry/Repository of Representation Information)
  - There is no single way in which the CPID is associated with a Data Object. CASPAR will have mechanisms that can be used by others, but we recognise that in order to have low-cost options, repositories must be able to use whatever mechanism is appropriate for its own systems. For example CASPAR will have a mechanism for packaging CPID with Data Objects, but a given repository may have a simple database which associates its internal Data Object name with the CPID.
- Persistent – the same identifier must continue to be resolvable over time even if the holder of that particular piece of Representation Information changes
- Multi-homed – by which we mean that a particular piece of RepInfo may be held in multiple Registries (as described in D1201)
  - Note that if a CPID enables a Data Object to be retrieved then these bytes are assumed to be the same whatever the particular source, apart from the Provenance and Fixity which may be associated with the Content Information. It would not be sensible to make mandatory checking this equality from all possible retrievable copies.
- Allow caching – a Repository must be able to keep copies of Representation Information within their own systems without having to change all CPIDs.
- Support versioning – we must allow the RepInfo to be updated – in order to allow sharing the effort of creating and maintaining RepInfo. Included in this is the conceptual digital object “the most recent”.
- Allow creation of new types of representation information.
- Reduce/remove dependencies on third party resolver systems.

1 Private communication
The work on CPIDs addresses two main areas:

1. definition and implementation of the CPID model shared by all the CASPAR components
2. definition of the CASPAR CPID usage for name resolution and resource discovery mechanism

5.3 IDENTIFICATION/RESOLUTION MODEL

5.3.1 Identification domain definition

The identification resource issue is connected to some general concepts that need to be considered during any identification, location and resolution processes.

Identifiers (IDs) are tokens that name entities. The concept is analogous to that of a name in the general common use. Identifiers are used extensively in virtually all information processing systems. Naming entities makes it possible to refer to them, which is essential for any kind of processing.

It is at least fortunate that, despite a variety of views on terminology, the basic concepts of sharing a common worldwide, universal identifier syntax - the URI syntax - has been fairly universally accepted. This notation provides an umbrella syntax that covers a variety of identifier strategies. The latest specification is available at http://www.ietf.org/rfc/rfc3986.txt.

5.3.2 URIs

As of 2007, there is consensus in the Identification community about relatively little: Internet, Web, digital library and scientific experts have offered a wide variety of designs for "improved" identification, beyond the defacto URL-based systems that have dominated since the rise of the Web. Even basic terminological concepts such as "URN", "URL", "URI" have proved surprisingly controversial.

In this document, we follow the terminology proposed in "URIs, URLs, and URNs: Clarifications and Recommendations 1.0" http://www.w3.org/TR/uri-clarification/ a Report from the joint W3C/IETF URI Planning Interest Group, adopting the so-called "Contemporary View".

Some relevant excerpt from this document:

There is some confusion in the web community over the partitioning of URI space, specifically, the relationship among the concepts of URL, URN, and URI. The confusion owes to the incompatibility between two different views of URI partitioning, which we call the "classical" and "contemporary" views.

Classical View

During the early years of discussion of web identifiers (early to mid 90s), people assumed that an identifier type would be cast into one of two (or possibly more) classes. An identifier might specify the location of a resource (a URL) or its name (a URN) independent of location. Thus a URI was either a URL or a URN. There was discussion about generalizing this by addition of a discrete number of additional classes; for example, a URI might point to metadata rather than the resource itself, in which case the URI would be a URC (citation). URI space was thus viewed as partitioned into subspaces: URL and URN, and additional subspaces, to be defined. The only such additional space ever proposed was URC and there never was any buy-in; so without loss of generality it is reasonable to say that URI space was thought to be partitioned into two classes: URL and URN. Thus for example, "http:" was a URL scheme, and "isbn:" would (someday) be a URN scheme. Any new scheme would be cast into one or the other of these two classes.

Contemporary View

Over time, the importance of this additional level of hierarchy seemed to lessen; the view became that an individual scheme does not need to be cast into one of a discrete set of URI types such as "URL", "URN", "URC", etc. Web-identifier schemes are in general URI schemes; a given URI scheme may define subspaces. Thus "http:" is a URI scheme. "urn:" is also a URI scheme; it
defines subspaces, called "namespaces". For example, the set of URNs of the form "urn:isbn:n-nn-nnnnnn-n" is a URN namespace. ("isbn" is an URN namespace identifier. It is not a "URN scheme" nor a "URI scheme").

Furthermore according to the contemporary view, the term "URL" does not refer to a formal partition of URI space; rather, URL is a useful but informal concept: a URL is a type of URI that identifies a resource via a representation of its primary access mechanism (e.g. its network "location"), rather than of URI space; rather, URL is a useful but informal concept: a URL is a type of URI that identifies a resource via a representation of its primary access mechanism (e.g. its network "location"), rather than by some other attributes it may have.

5.3.3 Identification vs Localization

As already mentioned, identifying a resource is assigning a unique name to the resource itself but this does not have anything to do with the actual resource physical (or logical) localization. Localization is the process of discovering the exact location (physical or digital) of a specific resource.

By its nature of being involved in long term data preservation, Caspar localization system should provide a stable but yet flexible way to localize any identified resource.

As explained before, URLs accomplish these tasks for digitally (and web) accessible resources but for their nature, URLs are strictly connected to digital and web resources so that we can not expect URLs to be used as the unique available formalism within the Caspar localization boundaries.

Up to now different formalisms tried to establish themselves as a standard for the identifying and localizing tasks, among them:

1. URLs
2. DOIs (http://en.wikipedia.org/wiki/Digital_object_identifier)
3. ARKs (http://www.cdlib.org/inside/diglib/ark/)

however it does not sound wise to limit Caspar localization tasks to just one of these formalisms, nor seems a good choice to prevent future formalisms definitions from being integrated into Caspar localization tools.

For all these reasons we shall allow multiple localization mechanisms to be used together, in the hope that at least one survives, in order to get to the host (or hosts) that holds the digital object or to somehow get the location of the identified physical object.

5.3.4 Model

With the entities defined, a sketch the model we need to have is as follows:
As identified in section 5.1.2 above there are resources within a CASPAR implementation that may be proposed for LSID (http://lsids.sourceforge.net/) could be more useful. The LSID concept introduces a straightforward approach to naming and identifying data resources stored in multiple, distributed data stores in a manner that overcomes the limitations of many naming schemes.

5.4 CASPAR RESOURCE IDENTIFIER CANDIDATE

As identified in section 5.1.2 above there are resources within a CASPAR implementation that may need identifiers but do not require such strict persistence requirements, for these a simpler system could be more useful.

Each system identifier is being expressed as a Universal Resource Name (URN). URN syntax is simple and extensible and under URN syntax a formalism that fits into Caspar needs is the one proposed for LSID (http://lsids.sourceforge.net/). The LSID concept introduces a straightforward approach to naming and identifying data resources stored in multiple, distributed data stores in a manner that overcomes the limitations of many naming schemes.

Following the LSID syntax constraints, a fully qualified CASPAR identifier consists of a ’CASPAR’ NID namespace identifier followed by an Authority identifier and by a specific Namespace suffixed by
an authority specific ObjectID. An optional additional Version section can be added to keep track of
data versioning:

\[ \text{URN:CASPAR:<Authority>:<Namespace>:<ObjectID>[::<Version>]} \]

The AUTHORITY string should be an Internet domain name which should be owned by the
organisation responsible to assign the URN identifier. This is normally a top-level domain (TLD)
registered to the organization, such as rl.ac.uk. Using separate authorities for each CASPAR digital
objects source can minimize the risk of collisions between entities of your data model and namespace
identifiers whilst allowing efficient allocation of new identifiers. The authority is responsible for the
uniqueness OBJECT-IDENTIFIER and persistence of the URN. The OBJECT-IDENTIFIER is a
unique (possibly) sequential string identifying a specific object inside the specified authority. No
location-dependent information should be included in the object identifier, instead a resolution service
should be used. To generate unique strings algorithms such as that used by the UUID
(http://www.dspx.net/uuid.html) are used. A UUID is a Universal Unique IDentifier a 128 bit number
that is guaranteed to be unique within the authority namespace. The mechanism used to guarantee
uniqueness is through combinations of hardware addresses, time stamps and random seeds.

![Diagram](image)

**Figure 26 Authority identifier use case**

By definition a CASPAR identifier identifies at most one resource, and can not be reassigned. Vice versa, a resource might have different identifiers referring to it, either inside the same authority (aliases) or into another authoritative namespace. It will then be advantageous to provide some kind of mapping or equivalence service for a resource.

### 5.4.1 Resource versioning, logical and unique identifiers

It is realistic to imagine that new versions of any preserved resource (or Representation Information
referred by Caspar) may be created from time to time, to improve usability or accuracy. The
versioning must be controlled and it will prove useful to distinguish between a unique identifier for a
particular version and a logical identifier for all versions of the preserved object. Using the logical
identifier should return the latest (and presumably the best) version, which will change as new
versions are created, whereas using the unique identifier, or, equivalently, providing a specific version
number, should always provide that specific identified resource.

### 5.4.2 Use of Extensible Resource Identifiers (XRI)

The status of the OASIS Extensible Resource Identifiers (XRI²) is, at the time of writing, that the XRI
Resolution 2.0 Committee Draft 02 is now in public review (running through 1 February 2008). The
XRI will not be discussed in detail here but it is clearly very flexible and in particular it delegates

---

identifier management not just in the authority segment (the first segment following the "xxx:/l" scheme name) but anywhere in the identifier path

5.5 CPID MODEL

Following from D1201 (section 7.3.2.2), conceptually we need to allow multiple name resolution mechanisms in the hope that at least one survives, in order to get to the host (or hosts) which hold the digital object.

An XML encoding may look something like:

```
<cpid>
   <!-- see below for further details -->
   <description>TBD</description>
   <value>xxxxxx</value>
   <locators>
      <locator locatorType="URL">http://x.y.z/a/b/c</locator>
      <locator locatorType="DOI">DOI:123456</locator>
      ...
   </locators>
</cpid>
```

We define:

- locators: a collection of individual locators
- locator: a string defining a specific path to locate (i.e. reach) a resource. A locator provides a pair: resolver name and resolver value which may provide an actionable way to locate a copy of an object. Any locator may be used.
  - locatorType: indicates the resolver to be used e.g. URL, DOI, ARK etc
  - value: optional string to be appended to each locator

5.5.1 CPID API Support

The software to support the CPID follows the general API described in Figure 24 where one expects to try each identifier until one works (i.e. not the protocol’s error return). In addition, to create an identifier one might expect automatic registration with a number of free resolver services (e.g. URL) plus any specified paid resolvers (e.g. DOI). Equality is determined by the equality of at least one locator.
6 INFORMATION DISCOVERY IN CASPAR

This Section describes the Discovery functionality of CASPAR, offered by the Finding Aids (FA from now on, for short) component. It is structured as follows: Section 6.1 discusses the requirements of FA identifying four basic requirements to be met by the corresponding component. A formalization of these requirements in terms of Use Cases has been included in the Architecture deliverable D1301 and is not replicated here. Instead, Section 6.2 presents a conceptual model meeting the identified requirements.

6.1 REQUIREMENTS FOR FINDING AIDS

The basic requirements for the CASPAR FA are those expressed in the OAIS Reference Model, recalled for self-containedness in section 6.1.1 below. Next, additional requirements are considered, imposed by the CASPAR conceptual model or by the technological context in which the project takes place. These requirements are exposed in Section 6.1.2. Section 6.1.3 finally summarizes the requirements.

6.1.1 Finding Aids and Descriptive Information in the OAIS Reference Model

According to OAIS, a “Finding Aid is an application that assists the Consumer in locating information of interest” 3. More specifically, a Finding Aid is a function of the Access functional area of the OAIS Reference Model, “a type of Access Aid that allows a user to search for and identify Archival Information Packages of interest”. Upon accessing an OAIS, “if the Consumer does not know a priori what specific holdings of the OAIS are of interest, the Consumer will establish a Search Session with the OAIS. During this Search Session the Consumer will use the OAIS Finding Aids that operate on Descriptive Information, or in some cases on the AIPs themselves, to identify and investigate potential holdings of interest. This may be accomplished by the submission of queries and the return of result sets to the Consumer”.

In the present context, we limit ourselves to FAs that operate exclusively on Descriptive Information through queries.

Descriptive Information is defined as “the set of information, consisting primarily of Package Descriptions, which is provided to Data Management to support the finding, ordering, and retrieving of OAIS information holdings by Consumers (…). Depending on the setting, this may be no more than a descriptive title of the Information Package that appears in some message, or it may be a full set of attributes that are searchable in a catalog service”. In the OAIS data model, Description Information is one type of Information Object (see taxonomy in Figure 27).

3 All quotations in this Section are from the OAIS Reference Model, unless otherwise specified.
As the conceptual modelling deliverable D1201 points out, “looking at existing archives one sees a very great variety of ACCESS-type functions. Indeed it is probably true to say that this, the user-facing part of an archive’s work, is the area in which the archive will seek to brand its services. Clearly the
access services have a certain degree of standardisation to allow interoperability, examples of which include provision of Web pages, OMI-PH harvesting, and FTP services. Nevertheless each archive will seek to provide a richer set of branded ordering, searching and data provision services”.

This requires the discovery functionality of CASPAR to be truly innovative, going beyond the common set of services nowadays found in most archives. On the other hand, it also requires to achieve a high degree of domain-independence in the type of discovery, something that could be exploited by many existing archives without incurring in an excessive cost.

The CASPAR conceptual modelling further suggests that “A type of Finding Aid which could have some discipline independent aspects is based on standardised PDI components, and in particular discipline independent aspects of Provenance, which CASPAR will be investigating in detail.”

From a more architectural point of view, the Architecture deliverable D1301 point out that “CASPAR does not aim to produce a general turnkey archive system”. Rather, the project will deliver a set of components which can be adopted and deployed independently from each other as much as possible, thus giving the freedom to adopters to select the installation that best suits their needs.

6.1.3 Discussion of requirements

In sum, the FA CASPAR component must support:

- the management of Descriptive Information,
- the association of Descriptive Information to IPs, and
- the querying of Descriptive Information for the discovery of the associated IPs

in a way that satisfies the following requirements:

- **data independence**, meaning that the component must not be bound to any specific format or type of Descriptive Information, thus reflecting the generality of the OAIS model. As for the format, some applications will find relational records adequate enough for describing the IPs they deal with, while other will want to be able to describe IPs in terms of sophisticated ontological descriptions; textual descriptions should also be possible. As for the type, some applications will want to discover IPs by stating queries on provenance, some other may rely on semantics or structure; the FA must be able to accommodate all these kinds, being agnostic to the type of used information.

- **query language independence**, meaning that the component must not be bound to any specific query language, thus reflecting the generality of the OAIS model also on this aspect. Clearly, the query language must be appropriate to the structure of the Descriptive Information it is used for, e.g. relational tables will have to be queried a relational language such as SQL.

- **representation adequacy**, meaning that the component must be able to manage not only logically simple descriptions but also very sophisticated ones, capturing much application semantics and possibly requiring logical inference for doing query evaluation.

- **architectural independence**, meaning that the component must offer a globally available service which is deployable as a stand-alone software system, not requiring any other CASPAR component for its usage. Technically speaking, this means that the FA component must itself be discoverable on a global infrastructure and self-contained, having on board all the required packages.

Requirements sometimes conflict, aggregating in inconsistent sets but this is not the case for the requirements listed above. This is due to the fact that current software and database technology makes it possible to achieve the required independencies in a relatively easy way. On the other hand, representational adequacy conflicts with efficiency (reasoning is in general a time-consuming task) and usability (some users are not at ease with logic). However, neither efficiency nor usability are considered to be strict requirements in this context, based on the consideration that a Designated
Community wishing to discover information on the basis of sophisticated descriptions is aware of the implications that such a choice has on efficiency and usability.

An effective way of meeting representation adequacy is to rely on a powerful knowledge representation language for Descriptive Information, allowing the user to create complex ontological descriptions, for instance dealing with the provenance of an IP. The discovery of IPs would then rely on engines that exploit such descriptions, as well as the involved ontologies, for query answering. This choice would allow the definition of FAs that are genuinely more powerful than the current available ones, mostly based on flat, record-like descriptions.

Semantic Web languages and the underlying technologies are natural candidates for playing this role.

6.2 A CONCEPTUAL MODEL OF FINDING AIDS

In order to meet the architectural requirement above, the FA is decomposed into two basic types of components:

1. Finding Registry and
2. Finding Manager.

- A Finding Registry supports the publication and discovery of Finding Managers, in the same way a UDDI server supports the publication and discovery of Web Services. In CASPAR. The concept of Finding Registry is introduced in order to be independent from any specific technology or de facto standard. Clearly, UDDI is a source of inspiration for Finding Registry and any UDDI implementation is a possible choice for implementing the Finding Registry; yet it is important to maintain these two notions separate and independent from each other.

In a CASPAR installation, there can be several Finding Registries, cooperating with each other in order to carry out their overall goal, possibly in a way that is transparent to applications; that is, an application needs not, and in general will not be aware of the existence of many Finding Registries that cooperate with the one the application interacts with.

Finding Registries follow authentication and authorization mechanisms as the rest of CASPAR components. This guarantees that only users with the appropriate roles will get access to Finding Aids functionality.

- A Finding Manager supports the management of Descriptive Information (DescInfo, hereafter for short), and is bound to a language for defining and for querying the managed DescInfo. For example, a Finding Manager may talk SQL, another one RDF/S, another one XML. This choice allows us to meet the data and query language independence requirements, and also supports implementation independence, since the same data and query language can have several implementations (SQL is a notable example). Moreover, it opens the way to meet the requirement of representation adequacy, since it makes it possible to define a Finding Manager based, say, on the ontology Web language OWL [ref].

Every Finding Manager registers with at least a Finding Registry in order to be discovered by applications.

The UML component diagram shown in Figure 28 illustrates the relations amongst the entities introduced so far within a CASPAR installation. Every CASPAR installation has at least a Finding Registry, while a Finding Registry can only serve a single installation. A Finding Registry registers at least one Finding Manager, while a Finding Manager may be registered with more than one Registry for reliability reasons. Every Finding Manager has exactly one Language, while several managers can talk the same language. Finally, every Manager manages exactly one DescInfo Repository (set of schema elements and associated objects), and vice versa a DescInfo Repository belongs to exactly one Manager.
6.2.1 Functional View

From a functional point of view, a Finding Registry supports three main functionalities:

1. Registration and Deregistration of a Finding Manager;
2. Access, Discovery and Browse of Finding Managers, and
3. Synchronization with the other Finding Registries of the same installation for the sharing of the information on the Finding Managers of the installation; this functionality is used for the proper functioning of the architecture and is not exposed to the applications.

A Finding Manager registers by providing a description of itself to the Registry. This description includes the following information:

- data definition and query languages spoken by the Finding Manager;
- details on how to invoke the Finding Manager (IP number of the server where the Finding Manager resides, protocol and the like);
- information concerning properties of the Finding Manager that applications consider useful for discovery purposes. This information may for instance refer to the content of the Finding Manager e.g., institution(s) contributing to the content, types of AIPs described by the Finding Manager and the like.

On the other hand, a Finding Manager supports two main functionalities:

1. Management of DescInfo:
   a. At the schema level: create, delete and browse DescInfo schemas.
   b. At the object level: create, delete, update and browse DescInfo objects.
2. Management of the association between DescInfo objects and AIP identifiers, including usage of these associations for AIP discovery:
   a. Create, delete query and browse association instances (i.e. (AIP-id, DescInfo-id) pairs).
   b. Discovery of AIPs via queries on DescInfo objects.

A user application interacts with the Finding Manager using a particular Query Language (QL) for querying and/or browsing the DescInfo Repository.

Below, Figure 29 sketch relations between above-mentioned Data Definition and Query languages, Descriptive Information schema and object, and Archival Information Packages.
6.3 HIGH LEVEL INTERFACE SPECIFICATION
Finding Aids components, Finding Manager and Finding Registry, offer three main interfaces to exhibit all required functionality.

Finding Registry access point is called ‘discoveryFM’, which is divided into two stereotypes:
(a) Access that allows register, unregister, and search Finding Manager components; and
(b) Manager that allows synchronization between different Finding Registries. See Figure 6.4, <<Interface>> FindingRegistry for methods signature.

Finding Manager access points are called ‘storeDescInfo’ and ‘discoveryAIP’ these and are divided into five stereotypes:
- storeDescInfo methods signatures are grouped into following stereotypes:
  o DescInfo schema management stereotype
  o DescInfo object management stereotype
  o AIP management stereotype
  o DescInfo-AIP management stereotype
- discoveryAIP methods signatures are grouped into following stereotype:
  o Access stereotype

The two remaining stereotypes from Figure 30 show FM initialization methods signature, i.e. to set the Data Definition Language and the Query Language.
6.4 LOGICAL VIEW

The logical view provides a hierarchical view of the component’s structure. It identifies the abstract characteristics of the objects (properties and features), and displays the entry points of the embedded code. According to the above sections Finding Aids is composed by two main components, Finding Registry and Finding Manager. In this Section, the Finding Manager is analyzed in detail, as it is the first one to be implemented according to the overall project plan. The Finding Registry will be developed in a second stage, thus we postpone its analysis.

6.4.1 Finding Manager Interfaces Refinement

Four main abstract classes form the Finding Manager logical component:

- FindingManager abstract class is responsible for FM configuration data and for query processing. It is also in charge of providing access to the Result Set.

- DescInfoSchemaManager abstract class provides the functionality to create, delete, and list DescInfo stored schemas.
DescInfoObjectManager abstract class manages the DescInfo object itself, which includes the association with the corresponding AIP.

DescInfoRepository abstract class is responsible for the FM storage. Three main concepts are stored: (a) DescInfo schemas, (b) DescInfo objects, (c) specific FM information, including AIP objects\(^4\) and their association with DescInfo Objects.

When a new Finding Manager is instantiated, a binding with a concrete Data Definition Language (DDL) and a concrete Query Language (QL) is established. All Descriptive Information managed by the FM is described and queried using them as well as FM internal data structures.

Target communities choose a FM instantiation based on DDL and QL. Once selected, (s)he creates a new schema using the DDL, in particular using the DescInfoSchemaManager interface to create, list, get and delete schema objects. Several communities use each FM to store its DescInfo schemas and objects.

DescInfoSchema and DescInfoSchema_ID abstract classes model DescInfo Schemas and its associated IDs.

DescInfoManager interface complete the DescInfo management by providing functionality to manage DescInfo objects and manage AIPs and manage DescInfo-AIP links. Both DescInfo objects management and AIP management include functionality to create, list, get and delete that objects. AIP – DescInfo links can be created and modified using this interface.

DescInfoObject and DescInfoObject_ID abstract classes model DescInfo Objects and its associated IDs. AIP objects are also modeled with FM (AIP and CASPAR_AIP_ID abstract classes) in order to keep necessary information, like AIP ID, but not the all AIP object.

Lastly, DescInfoRepository interface is a FM administration interface internally used by FM components to access to DescInfo (schemas, and objects) information.

\(^4\) Notice that in Finding Aids context an AIP object is not the true AIP but a representation of it that allows to identify the true AIP, i.e. its ID.
Figure 31 Finding Manager Logical View
To instantiate a concrete Finding Manager designer should implement it by extending described abstract classes. To do so a concrete Data Definition Language and a concrete Query language need to be chosen.

By fixing them the Finding Manager can manage different types of DescInfo objects, by storing one schema element for each type.

### 6.4.2 Finding Manager main sequence diagrams

Three main sequence diagrams are presented: the first is the standard FM startup sequence diagram, the second is the sequence corresponding to the storage of a concrete DescInfo (schema and object), and the third is to perform a query.

Main actor involved on FM start up (see Figure 32) is FM owner, i.e. institution that taking FM logical view with corresponding framework implementation instantiate a concrete FM by implementing DDL and QL operations.

CASPAR project provides an implementation of the FM that uses RDF Suite [reference], based on that context we describe the steps involved on Fin start up in following table:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FM creation. A new FM is initialized with a data definition language object and query language object.</td>
</tr>
<tr>
<td>2, 3, 4, 5, 6</td>
<td>Repository initialization. FM initialize its own repository to store its own data structures, DescInfo schemas and DescInfo objects.</td>
</tr>
<tr>
<td>7, 8, 9</td>
<td>DescInfo Object initialization. Thought these methods a new manager for DescInfo objects and AIPs is created.</td>
</tr>
<tr>
<td>10, 11, 12</td>
<td>DescInfo schema initialization. Thought these methods a new manager for DescInfo schemas is created.</td>
</tr>
<tr>
<td>15, 16</td>
<td>FM initialization. Finding manager data structures, e.g. to preserve DescInfo-AIP associations, are created and stored into repository.</td>
</tr>
</tbody>
</table>

Storing a new DescInfo object on a particular FM (see Figure 33) is performed by tree main steps:

- DescInfo schemaID: create a new schema for that object or get the schema ID of a previously stored schema.
- DescInfo object: create a new descriptive information object based on information provided as well as AIP object used from FM point of view. Associate these objects.
- Store: DescInfo object and association with AIP is stored on repository.

See table below:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3, 4</td>
<td>When a new AIP object is created and a DescInfo is provided using the DescInfoSchemaManager interface also a schema that represents that information is created and stored on the repository. If schema already exists the ones to get the schema ID replace these call.</td>
</tr>
<tr>
<td>5, 6, 7, 8</td>
<td>A new DescInfo object is created with provided information and an ID is assigned.</td>
</tr>
<tr>
<td>9, 10</td>
<td>A new (FM) AIP object is created with CASPAR AIP ID.</td>
</tr>
<tr>
<td>11, 12</td>
<td>Association between DescInfo object and AIP object is created and all information...</td>
</tr>
</tbody>
</table>
13  stored on repository.
14  Created DescInfo object ID is returned to caller.

Finding Manager main functionality is help on looking for AIP objects thought DescInfo querying. In order to do so at least two queries on Repository should be made from a logical point of view. One to get related DescInfo objects and other to get those associated AIP IDs. Let’s take a closer look on following steps from sequence diagram (see Figure 34):

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>A generic user (or application) submits a query by creating it and calling the discoveryAIP method.</td>
</tr>
<tr>
<td>3, 4</td>
<td>That query is appropriately parsed if necessary and passed to the RDF Suite to get the related items.</td>
</tr>
<tr>
<td>5, 6</td>
<td>The result set is parsed and relevant items that can be show to caller (DRM functionality) are used to retrieve all associated items (AIP IDs).</td>
</tr>
<tr>
<td>7, 8</td>
<td>AIP IDs query is performed.</td>
</tr>
<tr>
<td>9, 10</td>
<td>Result set is created and returned to caller.</td>
</tr>
</tbody>
</table>
6.4.3 Finding Manager start up

Figure 32 - FM start up
6.4.4 Finding Manager store DescInfo objects

Figure 33 - FM store DescInfo schema and object
6.4.5 Finding Manager discovery AIP

Figure 34 - FM search (AIPs)
6.5 IMPLEMENTING A FINDING MANAGER

This Section present a Finding Manager that is being implemented within the CASPAR project to support the discovery of AIPs according to the OAIS Reference Model and the conceptual model and interfaces presented so far.

Figure 35 shows the technological approach employed on the Finding Aids that involves three different levels: (a) Data layer, (b) Business layer, and (c) User Interface layer.

At a certain time a Finding Aids CASPAR component is deployed and configured to interact with the rest of a CASPAR installation, i.e. the minimum set of CASPAR components needed for digital preservation.

Test bed (artistic) partners provide digital objects and appropriate CASPAR tools (RepInfo Toolbox [reference!!]) aids Designated Community on creating appropriate Representation Information and the other necessary OAIS related information. Archival Information Packages (AIP) are created and stored on CASPAR system. By interacting with Finding Manager UI test bed partners associate specific Descriptive Information (DescInfo) to created AIPs.

Finding Aids logic allows storing DescInfo objects, the related AIP information, and other data relevant for preservation purposes. The Finding Aids support the user application in locating the relevant data. The user (application) is responsible to provide access to Finding Aids overall interface (see 6.2.1 - Functional View).

Consumers wants to search AIPs based on conditions upon the Descriptive Information, s/he specifies a certain Query and the Finding Aids answer with the corresponding Result Set based on stored DescInfo information and associated AIPs.

![Figure 35 – Finding Aids technological diagram (example with SWKM)](image)

Abstract classes and interfaces defined at business logic layer are instantiated as described below to implement Finding Manager Prototype, FM_WS version 0.1. This implementation is based on Java based web services (JAX-WS) deployed on a Glassfish container (UI layer), the open source
application server for Java EE. We first describe business logic layer and data layer implementation (Finding Manager based on SWKM (business and data layers)) and then we introduce UI layer by presenting FM web services.

6.5.1 Finding Manager based on SWKM (business and data layers)

Instantiation of this architecture uses Semantic Web Knowledge Manager (SWKM) [reference!] to preserve internal data structures as well as DescInfo object structure (graphspaces) and documents (namespaces). In this context the Data Definition Language is RDF [reference] and the Query Language is RQL [reference].

Finding Manager data structure graphspace that allows to keep updated the CASPAR AIP ID and the associated AIP is:

```xml
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
 <rdfs:Class rdf:ID="FindingManager"/>
 <rdfs:Class rdf:ID="AIP-ID"/>
 <rdfs:Class rdf:ID="DescInfo-ID"/>
 <rdfs:Property rdf:ID="DataDefinitionLanguage">
   <rdfs:domain rdf:resource="#FindingManager"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </rdfs:Property>
 <rdfs:Property rdf:ID="QueryLanguage">
   <rdfs:domain rdf:resource="#FindingManager"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </rdfs:Property>
 <rdfs:Property rdf:ID="associatedTo">
   <rdfs:domain rdf:resource="#DescInfo-ID"/>
   <rdfs:range rdf:resource="#AIP-ID"/>
 </rdfs:Property>
</rdf:RDF>
```

Table 1 - association CASPAR AIP ID-DescInfo using RDFS

Let's describe "implementation" classes (light blue classes from Figure 36 - Finding Manager instantiation with SWKM), which are responsible for FM business logic and SWKM link:

- FindingManagerImpl class is responsible for FM initialization and provides access to discovery of AIPs.
  - initialize method create the new DB used by SWKM, and it uses the fmClient.importer().store ( col, Deps.WITHOUT ) method to store RDF schema on DB.
  - discoveryAIP method query SWKM and return related DescInfo Objects and the associated CASPAR AIP IDs by using fmClient.query().query ( Format.RDF.XML, query ) call.

- DescInfoSchemaManagerImpl class manages named schemas that contain DescInfo schemas and FM schemas (data structure). Implemented methods are:
o createDescInfoSchema method stores a new DescInfo named schema by calling the SWKM method client.importer().store ( col, Deps.WITHOUT).

o deleteDescInfoSchema method do nothing because delete schema elements is not supported into SWKM current implementation.

o getDescInfoSchema

o listDescInfoSchema

- DescInfoobjectManagerImpl class is responsible for create, delete, get, and list DescInfo and AIP objects (both are named graphs) mapping this method as described for DescInfoSchemaManagerImpl class. Moreover it is responsible for associations between CASPAR AIP ID and DescInfo objects by implementing following methods:
  o associateDescInfoToAIP
  o disassociateDescInfoToAIP
  o getAssociatedDescInfo

- DescInfoRepositoryImpl class is used by FM classes in order to directly access to DescInfo repository. Provided functionality allows to list, get, and delete stored objects as well as store a new one or update and existing object, ie. to update: 
  fnClient.update().update( 
  updateStatement ), and to get the contents of a given namespaces: 
  client.exporter().fetch(Arrays.asList ( namespace ), Format.RDF_XML, 
  Deps.WITHOUT, Data.WITHOUT).

Lastly, the required software components to deploy and run FM with SWKM are:

- Java EE 5
- Sun Java System Application Server Platform Edition 9.0_01 (build b02-p01) – shortly called “Glassfish”
- The SWKM web application (Java) deployed in the servlet container (or the application server). See SWKM documentation for other requirements.

### 6.5.2 Web Service interface for the Finding Manager (UI layer)

The Web Service interface for the Finding Manager, FM_WS, is packaged in the form of a single WAR (Web Application aRchive) file and contains the described logical model using RDF Suite as described before and the Web service interface.

The FM_WS war file is follows the “CASPAR COMPONENTS BEST PRACTICES” guidelines document. Basically, it contains 4 java packages:

1. eu.casparpreserves.impl.findingaids, this package contains the implementation of the business logic of the CASPAR FM that uses SWKM. No dependencies with the web service technology and the CASPAR framework is present here;

2. eu.casparpreserves.skeleton.findingaids, contains all web service classes that answer web clients calls, that classes uses business logic classes to satisfy clients;

3. eu.casparpreserves.stub.findingaids, all stub classes used by the remote client to send calls to the are in this package;

4. eu.casparpreserves.test.findingaids, test client classes are in this package.

The scope of this organization is to divide the business logic (business layer) from the web service technologies (UI layer). In fact, it is possible to change the web service technologies, with minor consequences on the Finding Manager business and data layers. This is an kind of “preservation”, that we called “source code preservation”.
Required software components to deploy and run FM_WS are:
- Java EE 5 (http://java.sun.com)
- Glassfish V2 (https://glassfish.dev.java.net/)
- JAX_WS 2.1.2 (https://jax-ws.dev.java.net/2.1.2/)
- The FM_WS web application (Java) deployed in the servlet container (or the application server).

FM_WS web services can be accessed through a standard WSDL interfaces. Accessing the services themselves after deployment like can retrieve the corresponding WSDL descriptions of FM_WS services:

http://your.server.ip/CASPAR_FindingManager_WS/ServiceName?wsdl

For testing and integration purposes a working installation id provided by CNR at:

In order to facilitate interactions with the server functionality a client example is provided, FindingManagerClient class. It creates a proxy from stub classes to gain access to FM and shows a set of examples of how server can be used to store DescInfo objects and discover AIPs.
Figure 36 - Finding Manager instantiation with SWKM
7 CONCLUSIONS

The components presented here touch upon all aspects of the access to the digitally encoded information, as defined by the CASPAR Conceptual Model and Architecture.

The level of detail presented here differs from one area to another, as each presents its own particular issues, some of which require considerable background conceptual investment whereas others are more technology centred. Preservation, underpinned by the OAIS Reference Model, provides the unifying principles. We have tried for all these aspects, except perhaps for the Finding Aids, to be sure that the artefacts which are produced, such as during the capture of Provenance, are themselves preservable. In doing this we apply the same principles which we apply to any other Data Object.

We recognise that the unavoidable recursion which this implies needs to be treated carefully but we believe that as long as we keep in mind that we need to deal with all types of digital objects then this is not an insuperable problem.

As work progresses in CASPAR, prototypical implementations of the corresponding components will be delivered according to the schedule outlined in the Architecture Deliverable, following the CASPAR iterative methodology.