PREMIS Data Dictionary for preservation metadata

Version 3.0 -- June 2015 -- Revised November 2015
(based on pdf version at https://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf)

Prepared and maintained by the

PREMIS Editorial Committee

PREMIS Data Dictionary for preservation metadata, Version 3.0 -- June 2015 -- Revised November 2015 (based on pdf version at https://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf)

Available from
Library of Congress, Network Development & MARC Standards Office
101 Independence Ave., SE
Washington, DC 20540
USA
ndmso@loc.gov

Edition : November 2015 Edition
Printed : Printed in the United States of America

PLACEHOLDER ISBN VALUE

Table of Contents

ACKNOWLEDGMENTS

PREMIS Editorial Commttee Members

Rebecca Guenther, Library of Congress, Chair
Karin Bredenberg, Riksarkivet, Swedish National Archives
Angela Dappert, University of Portsmouth
Angela Di Iorio, Sapienza Università di Roma
Leslie Johnston, U.S. National Archives and Records Administration
Devon Landes, HBO
Peter McKinney, National Library of New Zealand
Evelyn McLellan, Artefactual Systems
Tracy Meehleib, Library of Congress
Sébastien Peyrard, Bibliothèque nationale de France
Pauline Sinclair, Preservica
Eld Zierau, Royal Library of Denmark

Special Thanks

The following contributed their expertise to previous versions as former members of the PREMIS Editorial Committee:

Steve Bordwell, General Register Office for Scotland
Yair Brama, ExLibris
Olaf Brandt, Koninklijke Bibliotheek, Netherlands
Priscilla Caplan, Florida Center for Library Automation (co-chair of original PREMIS Working Group)
Gerard Clifton, National Library of Australia
Markus Enders, British Library
Noreen Hill, Library and Archives Canada
Karsten Huth, Sächsisches Staatsarchiv, Saxon State Archives
David Lake, U.S. National Archives and Records Administration
Brian Lavoie, OCLC
Yaniv Levi, ExLibris
Bill Leonard, Library and Archives Canada
Rory McLeod, British Library
Robert Sharpe, Preservica
Robert Wolfe, HBO
Zhiwu Xie, Los Alamos National Laboratory
Sally Vermaaten, Statistics New Zealand
Kate Zwaard, U.S. Government Printing Office, Library of Congress

In addition to Editorial Committee members, the following contributed their expertise to the work on the Environment Working Group for PREMIS 3:

Conçalo Antunes, Instituto de Engenharia de Sistemas e Computadores
Artur Caetano, Instituto de Engenharia de Sistemas e Computadores
Carol Chou, Florida Virtual Campus
Janet Delve, University of Portsmouth
Martin Neumann, University of Karlsruhe
Michael Nolan, Intel

In addition to Editorial Committee members, the following contributed their expertise to the work on the Conformance Statement for PREMIS 3:

Jay Gattuso, National Library of New Zealand
Jan Hutař, Archives New Zealand
Amy Kirchhoff, ITHAKA
Joseph Pawletko, New York University

The following people were the original Preservation Metadata: Implementation Strategies (PREMIS) Working Group that developed version 1 of the Data Dictionary:

Priscilla Caplan, Florida Center for Library Automation, co-chair
Rebecca Guenther, Library of Congress, co-chair
Robin Dale, RLG liaison
Brian Lavoie, OCLC liaison
George Barnum, U.S. Government Printing Office
Charles Blair, University of Chicago
Olaf Brandt, Göttingen State and University Library
Mikki Carpenter, Museum of Modern Art
Adam Farquhar, British Library
David Gewirtz, Yale University
Keith Glavash, MIT/DSpace
Andrea Goethals, Florida Center for Library Automation
Cathy Hartman, University of North Texas
Helen Hodgart, British Library
Nancy Hoebelheinrich, Stanford University
Roger Howard, J. Paul Getty Museum
Sally Hubbard, Getty Research Institute
Mela Kircher, OCLC
John Kunze, California Digital Library
Vicky McCargar, Los Angeles Times
Jerome McDonough, New York University/METS
Evan Owens, Ithaka-Electronic Archiving Initiative
Erin Rhodes, U.S. National Archives and Records Administration
Madi Solomon, Walt Disney Corporation
Angela Spinazze, ATSPIN Consulting
Stefan Strathmann, Göttingen State and University Library
Günter Waibel, RLG
Lisa Weber, U.S. National Archives and Records Administration
Robin Wendler, Harvard University
Hilde van Wijngaarden, National Library of the Netherlands
Andrew Wilson, National Archives of Australia and British Library
Deborah Woodyard-Robinson, British Library and Woodyard-Robinson Holdings Ltd.

PREMIS WEBSITES & EMAILS

PREMIS maintenance activity Web site: http://www.loc.gov/standards/premis/.

PREMIS Implementers' Group discussion list: pig@loc.gov. To subscribe, send an e-mail to listserv@loc.gov with the message, "subscribe pig [your name]".

Please send comments and questions to premis@loc.gov.

INTRODUCTION

The PREMIS Data Dictionary is a comprehensive, practical resource for implementing preservation metadata in digital preservation systems. The Data Dictionary defines preservation metadata that:

- Supports the viability, renderability, understandability, authenticity, and identity of digital objects in a preservation context;
- Represents the information most preservation repositories need to know to preserve digital materials over the long term;
- Emphasizes “implementable metadata”: rigorously defined, supported by guidelines for creation, management, and use, and oriented toward automated workflows; and,
- Embodies technical neutrality: no assumptions made about preservation technologies, strategies, metadata storage and management, etc.

Background

Development of the original PREMIS Data Dictionary

In June 2003, OCLC and RLG jointly sponsored the formation of the PREMIS (***Pre****servation* ***M****etadata:* ***I****mplementation* ***S****trategies*) working group, comprised of international experts in the use of metadata to support digital preservation activities. The working group's membership included more than 30 participants, representing five different countries and a variety of domains, including libraries, museums, archives, government agencies, and the private sector. Part of the working group's charge was to develop a core set of implementable preservation metadata, broadly applicable across a wide range of digital preservation contexts and supported by guidelines and recommendations for creation, management, and use. This portion of the working group's charge was fulfilled in May 2005 with the release of *Data Dictionary for Preservation Metadata: Final Report of the PREMIS Working Group*. In addition to the Data Dictionary, the working group also published a set of XML schemas to support implementation of the Data Dictionary in digital preservation systems.

The PREMIS working group was established to build on the earlier work of another initiative sponsored by OCLC and RLG: the Preservation Metadata Framework (PMF) working group. In 2001–2002 the PMF working group outlined the types of information that should be associated with an archived digital object. Their report, *A Metadata Framework to Support the Preservation of Digital Objects* (the *Framework*), proposed a list of prototype metadata elements.[1]However, additional work was needed to make these prototype elements implementable. The PREMIS working group was asked to take the PMF group's work a step further and develop a data dictionary of core metadata for archived digital objects, as well as give guidance and suggest best practice for creating, managing, and using the metadata in preservation systems.

Since the PREMIS working group had a practical rather than theoretical focus, members were sought from institutions known to be operating or developing preservation repository systems within the cultural heritage and information industry sectors. Diverse perspectives were also sought. The working group consisted of representatives from academic and national libraries, museums, archives, government, and commercial enterprises in five different countries. In addition, PREMIS called upon an international advisory committee of experts to review progress.

To understand how preservation repositories were actually implementing preservation metadata, in November 2003 the working group undertook a survey of about 70 organizations thought to be active in or interested in digital preservation, resulting in the report *Implementing Preservation Repositories*

1 A Metadata Framework to Support the Preservation of Digital Objects (Dublin, Ohio: OCLC Online Computer Library Center, 2002), http://www.oclc.org/research/projects/pmwg/pm_framework.pdf.

for Digital Materials: Current Practice and Emerging Trends in the Cultural Heritage Community (the *Implementation Survey Report*).[2] The findings of this survey were extremely helpful in informing the working group's discussions as it developed the Data Dictionary.

2 *Implementing Preservation Repositories for Digital Materials: Current Practice and Emerging Trends in the Cultural Heritage Community* (Dublin, Ohio: OCLC Online Computer Library Center, 2004), http://www.oclc.org/research/projects/pmwg/surveyreport.pdf

Implementable, core preservation metadata

Both the earlier *Framework* and the PREMIS Data Dictionary build on the Open Archival Information System (OAIS) reference model (ISO 14721).[3] The OAIS information model provides a conceptual foundation in the form of a taxonomy of information objects and packages for archived objects, and the structure of their associated metadata. The *Framework* can be viewed as an elaboration of the OAIS information model, explicated through the mapping of preservation metadata to that conceptual structure. The PREMIS Data Dictionary can be viewed as a translation of the *Framework* into a set of implementable semantic units. However, it should be noted that the Data Dictionary and OAIS occasionally differ in terminology usage; these differences are noted in the Glossary that accompanies this report. Differences usually reflect the fact that PREMIS semantic units require more specificity than the OAIS definitions provide, which is to be expected when moving from a conceptual framework to an implementation.

The Data Dictionary defines "preservation metadata" as *the information a repository uses to support the digital preservation process*. Specifically, the group looked at metadata supporting the functions of maintaining viability, renderability, understandability, authenticity, and identity in a preservation context. Preservation metadata thus spans a number of the categories typically used to differentiate types of metadata: administrative (including Rights and permissions), technical, and structural. Particular attention was paid to the documentation of digital provenance (the history of an object) and to the documentation of relationships, especially relationships among different objects within the preservation repository. Version 3.0 of the Data Dictionary explicitly expands the scope beyond repository boundaries in order to accommodate seamless metadata representation across the digital object life-cycle.

The group considered a number of definitions of "core." In one view, core describes any metadata absolutely required under any circumstances. In another, core means that metadata is applicable to any type of repository implementing any type of preservation strategy. PREMIS uses this practical definition: *things that most working preservation repositories are likely to need to know in order to support digital preservation*. The words "most" and "likely" were chosen deliberately. Core does not necessarily mean mandatory, and some semantic units were designated as optional when exceptional cases were apparent.

The concept of "implementability" also required definition. Most preservation repositories deal with large quantities of data. Therefore, a key factor in the implementability of preservation metadata is whether the values can be automatically supplied and automatically processed by the repository. Whenever possible the group defined semantic units that do not require human

3 ISO 14721:2012: Space Data and information transfer system—Open archival information system (OAIS)—Reference model (OAIS) (Geneva, Switzerland: International Organization for Standardization, Aug. 2012).

intervention to supply or analyze. For example, controlled values from an authority list are preferred over textual descriptions.

The working group decided that the Data Dictionary should be wholly implementation independent. That is, the core metadata define information that a repository needs to know, regardless of how, or even whether, that information is stored. For instance, for a given identifier to be usable, it is necessary to know the identifier scheme and the namespace in which it is unique. If a particular repository uses only one type of identifier, the repository would not need to record the scheme in association with each object. The repository would, however, need to know this information and to be able to supply it when exchanging metadata with other repositories. Because of the emphasis on the need to know rather than the need to record or represent in any particular way, the group preferred to use the term “semantic unit” rather than “metadata element.” The Data Dictionary names and describes semantic units.

PREMIS Maintenance Activity

Following the release of the Data Dictionary in 2005, the PREMIS working group retired and the PREMIS Maintenance Activity, sponsored by the Library of Congress, was initiated to maintain the Data Dictionary and coordinate other work to advance understanding of preservation metadata and related topics. In addition to providing a permanent Web home for the Data Dictionary, XML schema, and related materials, the Maintenance Activity also operates the PREMIS Implementers Group (PIG) discussion list and wiki, conducts tutorials on the Data Dictionary and its use, and commissions focused studies on preservation metadata topics. The Maintenance Activity also established an Editorial Committee responsible for further development of the Data Dictionary and the XML schema and promoting their use. Themembership of the Editorial Committee reflects a variety of countries and institutional backgrounds.

Users identify errors, and provide feedback on ways that the Data Dictionary could be improved to increase its value and ease of application through a variety of mechanisms: Discussion of issues takes place on the PREMIS Implementers Group (PIG) discussion list and wiki, user group meetings take place at PREMIS Implementation Fairs that are organized in conjunction with major conferences and are advertised on the PREMIS website and through mailing lists, and requests for changes can be submitted through the *PREMIS Data Dictionary and Schema Revision Process* specified at http://www.loc.gov/standards/premis/revision-process.html.

The members of the Editorial Committee revise the Data Dictionary when a sufficient level of commentary has accumulated to warrant doing so, making every effort to engage stakeholders in the process of revision. The Committee keeps the preservation community informed of issues being discussed, solicits comment on proposed revisions, and consults outside experts where appropriate.

Version History

Version	Release Date
Version 1.0	May 2005
Schema Version 1.0	*17 May 2005*
Schema Version 1.1	*27 September 2005*
Version 2.0	March 2008
Schema Version 2.0	*17 July 2008*
Version 2.1	January 2011
Schema Version 2.1	*6 January 2011*
Version 2.2	July 2012
Schema Version 2.2	*15 May 2012*
Schema Version 2.3	*4 August 2014*
Version 3	June 2015

Subsequent versions of the Dictionary have taken advantage of the increase in use of PREMIS and experience of implementing preservation solutions. They have included corrections of errors, clarifications of some semantic units, changes for consistency, and the addition of a few semantic units that resulted from requests to the PREMIS Editorial Committee. Version 2.0 was a major revision and version 2.1 added additional functionality, particularly in the Rights entity. Version 2.2 was considered non-substantial as there were no major changes affecting existing PREMIS descriptions. Version 2.3 contained only changes that affected the XML schema (by adding the ability to designate use of specific controlled vocabularies); these were not reflected in the Data Dictionary. Further information about this mechanism is given under the section “Supplying Metadata Values”. Starting from version 2.2, a PREMIS OWL ontology is available alongside the XML Schema. It allows one to provide a PREMIS-endorsed flavor of the Data Dictionary so that one can express preservation metadata in RDF. This ontology does not replace but complements XML in areas where RDF may be better suited, such as querying or publishing preservation metadata, or connecting repository-specific data to externally maintained registries.

This version of the Data Dictionary, version 3, includes some major changes and additions to the Dictionary. These can be highlighted as:

- Repositioning of Intellectual Entity as a category of Object to enable additional description within PREMIS and linking to related PREMIS entities.
- Repositioning of Environments (i.e. hardware and software needed to use digital objects) so that they can be described and preserved reusing the Object entity. That is to say, they can be described as Intellectual Entities and preserved as Representation, File or Bitstream Objects.
- Addition of physical Objects to the scope of PREMIS so that they can be described and related to digital objects.

- Addition of a new semantic unit to the Object entity: preservationLevelType (O, NR) to indicate the type of preservation functions expected to be applied to the object for the given preservation level.
- Addition of a new semantic unit to the Agent entity to express the version of software Agents: agentVersion (O, NR).
- Addition of a new semantic unit to the Event entity: eventDetailInformation (O, R).

Major additions are discussed in detail below (see the “PREMIS Data Model” and “Environment” sections). Other additions are explained within the relevant section of the Dictionary.

PREMIS Awards and Recognition

The PREMIS Data Dictionary was awarded the 2005 Digital Preservation Award (given under the auspices of the British Conservation Awards), the 2006 Society of American Archivists Preservation Publication Award, and was a finalist for the 2012 Digital Preservation Award for the most outstanding contribution to digital preservation in the last decade.

The PREMIS Data Model

The PREMIS Data Dictionary defines **semantic units**. Each semantic unit defined in the Data Dictionary is mapped to an entity that is organized within a simple data model. A semantic unit can therefore be understood as a property of an entity. The model defines four entities important in regard to digital preservation activities: Objects, Events, Agents and Rights.[4] Figure 1 provides a graphical illustration of the PREMIS Data Model.

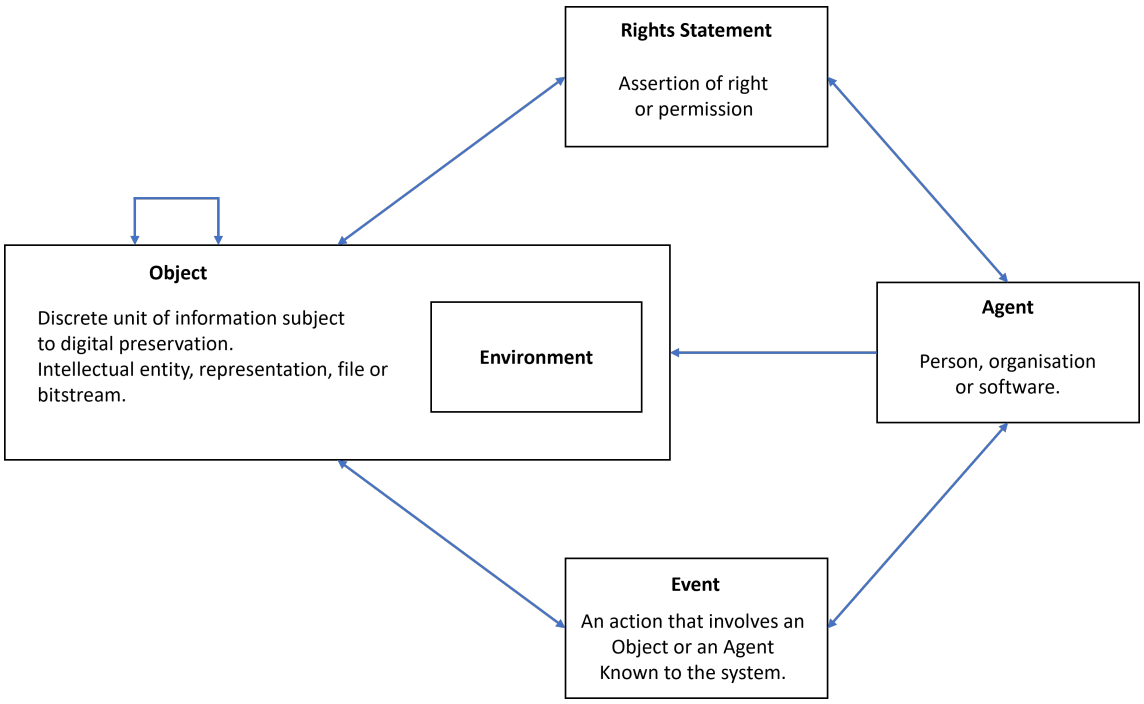

Figure 1. The PREMIS Data Model

In Figure 1, entities are represented by boxes; relationships between entities are represented by arrows. When arrows are bi-directional, then each entity type contains a semantic unit allowing it to link to the other. So, for example, the Rights entity includes a semantic unit recording information about the relationship with an Agent, and the Agent entity includes a semantic unit recording information about associated Rights.

The arrow pointing from the Objects entity back to itself indicates that the semantic units defined in the Data Dictionary support the recording of relationships between Objects. No other entity in the data model supports relationships of this type; in other words, while Objects can be related to other Objects, Events cannot be related to other Events, Agents cannot be related to other Agents, and so on.

4 Other preservation metadata initiatives have developed other models. The National Library of New Zealand defines four types of entity: objects, files, processes, and metadata modification. Metadata Standards Framework—Preservation Metadata (Revised) (Wellington: National Library of New Zealand, June 2003), http://digitalpreservation.natlib.govt.nz/assets/Uploads/nlnz-data-model-final.pdf.

The entities in the PREMIS data model are defined as follows:

Object (or Digital Object): a discrete unit of information subject to digital preservation.[5] Version 3 introduces the notion that this can be an environment used as part of the preservation process.

Environment: Technology (software or hardware) supporting a Digital Object in some way (e.g. rendering or execution). Environments can be described as Intellectual Entities and captured and preserved in the preservation repository as Representations, Files and/or Bitstreams.

Event: an action that involves or affects at least one Object or Agent associated with or known by the preservation repository.

Agent: person, organization, or software program/system associated with Events in the life of an Object, or with Rights attached to an Object. It can also be related to an environment Object that acts as an Agent.

Rights Statement: assertion of one or more Rights or permissions pertaining to an Object and/or Agent.

Each semantic unit defined in the Data Dictionary is mapped to one of the entities in the data model. In this sense, a semantic unit may be viewed as a property of an entity. For example, the semantic unit *size* is a property of an Object entity. Semantic units have values: for a particular object the value of *size* might be "843200004."

In most cases, a particular semantic unit is unambiguously a property of only one type of entity. The size of an object is clearly a property of the Object entity. In some cases, however, a semantic unit applies equally to two or more types of entity. For example, Events have outcomes, which could be a property of the Event or the Object or Agent affected. If a migration eventcreates a file that has lost some important feature, the loss of that feature might be considered an outcome of the event, and therefore a property of the Event entity. Alternatively, it might be considered an attribute of the new file, and therefore a property of the Object entity. When a semantic unit applies equally to multiple entity types, the decision has been taken that the semantic unit should be associated with only one type of entity in the Data Dictionary. The data model relies upon links between the different entities to make these relationships clear. In the example above, the loss of the feature is treated as a detailed outcome of the Event, where the Event contains the identifier of the Object involved. What is important is that this association is arbitrary and is not meant to imply that a particular implementation is required. The choice of semantic unit is down to individual implementations.

5 Note that the PREMIS definition of an Object entity differs from the definition of digital object commonly used in the digital library community, which holds a digital object to be a combination of identifier, metadata, and data. This is not intended to be a conflict. The Object entity in our model is an abstraction defined only to cluster attributes (semantic units) and clarify relationships.

In some cases a semantic unit takes the form of a **container** that groups a set of related semantic units. For example, the semantic unit *identifier* groups the two semantic units *identifierType* and *identifierValue*. The grouped subunits are called **semantic components** of the container. Some containers are defined as **extension containers**, to allow the use of metadata encoded according to an external schema. This enables PREMIS to be extended with metadata elements that are more granular, non-core, or otherwise out of scope for the Data Dictionary.

A **relationship** is a statement of association between instances of entities. "Relationship" can be interpreted broadly or narrowly, and expressed in many different ways. For example, the statement "Object A is of format B" could be considered a relationship between A and B. The PREMIS model, however, treats format B as a property of Object A. PREMIS reserves "relationship" for associations between two or more Object entities or between entities of different types, such as an Object and an Agent.

More on Objects

The Object entity has four subcategories: Intellectual Entity, Representation, File, and Bitstream.

An **Intellectual Entity** is a distinct intellectual or artistic creation that is considered relevant to a designated community in the context of digital preservation: for example, a particular book, map, photograph, database, or hardware or software. An Intellectual Entity can include other Intellectual Entities; for example, a web site can include a web page and a web page can include an image. An Intellectual Entity may have one or more digital or non-digital Representations.

A **Representation** is the set of files, including structural metadata, needed for a complete rendition of an Intellectual Entity. For example, a journal article may be complete in one PDF file; this single file constitutes the Representation. Another journal article may consist of one SGML file and two image files; these three files constitute the Representation. A third article may be represented by one TIFF image for each of 12 pages plus an XML file of structural metadata showing the order of the pages; these 13 files constitute the Representation. Starting with PREMIS version 3.0 physical items, such as manuscripts or printed documents, may also be Representations so that digital and non-digital Representations can be captured uniformly.

A **File** is a named and ordered sequence of bytes that is known to an operating system. A File can be zero or more bytes and has a File format, access permissions, and File system characteristics such as size and last modification date.

A **Bitstream** is contiguous or non-contiguous data within a file that has meaningful common properties for preservation purposes. A Bitstream cannot be transformed into a standalone file without the addition of file structure (headers, etc.) and/or reformatting the Bitstream to comply with some particular file format.

The relationship between the subcategories is illustrated in Figure 2. The subcategories are discussed in more detail below.

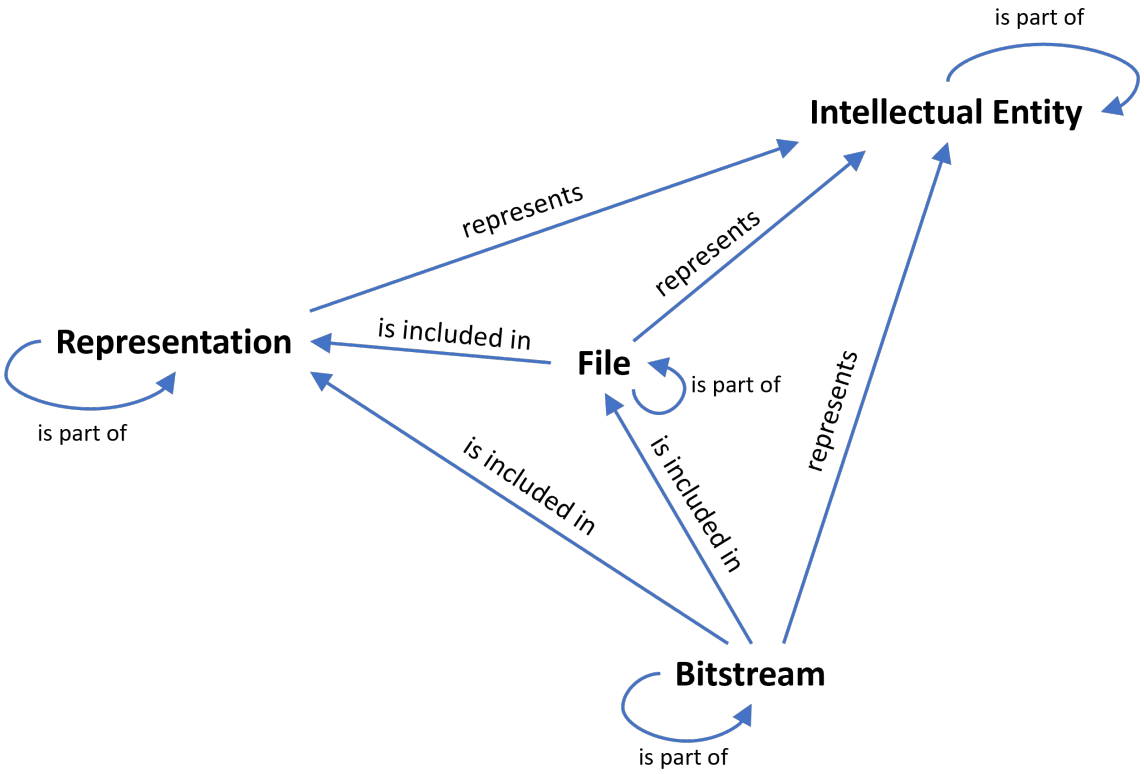

Figure 2. Conceptual view between object categories

Intellectual Entities

An **Intellectual Entity** is a distinct intellectual or artistic creation that is considered relevant to a designated community in the context of digital preservation. An Intellectual Entity can include other Intellectual Entities and it may have one or more Representations.

Before versions 3.0 of the Data Dictionary Intellectual Entities were considered out of scope for PREMIS, since many types of Intellectual Entities--such as books, articles, images, archival collections, or statistical studies – do not necessarily need to be described as part of the preservation metadata and are well-served by descriptive metadata that supports their discovery and retrieval from outside the preservation metadata.

In version 3.0, one now has the choice of modeling Intellectual Entities outside PREMIS or within it.

Consider an example where the Intellectual Entity is specified outside PREMIS (as has been the case in previous versions). A TIFF Representation for an image has the identifier “galapagos.6754.1”. Note that instead of using the *linkingIntellectualEntityIdentifier* from version 2, the Representation now links to the externally specified Intellectual Entity using a structural “represents” relationship with a *relatedObjectIdentifier* whose value is an external actionable HTTP URI identifier. There may be additional Representations, such as a jpg image or a thumbnail, relating to the same Intellectual Entity. In this case no explicit PREMIS Object is created for the Intellectual Entity.

premis:objectIdentifier
  premis:objectIdentifierType = "hdl"
  premis:objectIdentifierValue = "galapagos.6754.1"
premis:objectCategory = representation
premis:relationship
  premis:relationshipType = "structural"
  premis:relationshipSubType = "represents"
  premis:relatedObjectIdentifier
    premis:relatedObjectIdentifierType = "URI"
    premis:relatedObjectIdentifierValue = "http://natureweb/col1.galapagos6754/default.html"

Alternatively, the Intellectual Entity can be modeled as an autonomous Object within PREMIS with objectCategory "Intellectual Entity". PREMIS implementation experience and the modeling work of the Planets Project[6] have shown that repositories may have a need to record descriptive metadata as well as preservation information, such as significant properties, relationships, rights, and related events information, at the Intellectual Entity level.

6 Report on policy and strategy models for libraries, archives and data centres, July 2009, http://www.planets-project.eu/docs/reports/Planets_PP2_D3_ReportOnPolicyAndStrategyModelsM36_Ext.pdf.

In this second example, the Intellectual Entity is instead specified within PREMIS. Once again, aTIFF Representation for an image has the identifier “galapagos.6754.1”. However, this time its descriptive metadata is captured in an Intellectual Entity instance. The Intellectual Entity’s identifier within PREMIS is “col1.galapagos6754”, a stable handle identifier, which is persistent and suited for linking within the repository. Additionally, the Intellectual Entity has an actionable identifier (HTTP URI), which links to its catalog record. Furthermore it records the Event (E004) when the Intellectual Entity was added to the digital collection. The repository creates separate records for the Representation, the Intellectual Entity, and the related Event:

For the image Representation linking to its Intellectual Entity in PREMIS:

premis:objectIdentifier
  premis:objectIdentifierType = "hdl"
  premis:objectIdentifierValue = "galapagos.6754.1"
premis:objectCategory = representation
premis:relationship
  premis:relationshipType = "structural"
  premis:relationshipSubType = "represents"
  premis:relatedObjectIdentifier
    premis:relatedObjectIdentifierType = "hdl"
    premis:relatedObjectIdentifierValue = "col1.galapagos6754"

For its Intellectual Entity in PREMIS recording both identifiers and the Event:

premis:objectIdentifier
  premis:objectIdentifierType = "hdl"
  premis:objectIdentifierValue = "col1.galapagos6754"
premis:objectCategory = intellectual entity
premis:objectIdentifier
  premis:objectIdentifierType = "URI"
  premis:objectIdentifierValue = "http://natureweb/col1.galapagos6754/default.html"
premis:objectCategory = intellectual entity
premis:linkingEventIdentifier
  premis:linkingEventIdentifierType = "Local
  Repository"premis:linkingEventIdentifierValue = "E004"

Intellectual Entities occur at all levels of aggregation from collections down to individual embedded images. They were introduced because somerepositories have a need to describe sets of content containing multiple Representations that are primarily aggregated for preservation purposes. For example, an Intellectual Entity may be used to record information about packages containing multiple Representations that are used for inter-repository exchange (e.g. the TIPR Repository Exchange Format[7]). Outputs from complex Events such as web crawls may also be represented as Intellectual Entities. To

7 TIPR: Towards Interoperable Digital Repositories. Repository eXchange Package (RXP) Spec, Version 1.0. http://wiki.fcla.edu/TIPR/21.

support these and other use cases, Intellectual Entities are now treated as a category of Object in the PREMIS Data Model. The metadata in PREMIS for Intellectual Entities includes only that which supports the preservation process and not full descriptive metadata required for discovery. External metadata schemes are used for this purpose, and there are various methods to relate the PREMIS Object to its descriptive metadata (e.g. a link to a catalog record as illustrated above; descriptive metadata in a METS[8] document; links between a preservation repository and another system).

Use of Intellectual Entities needs to be adapted to the stakeholder's use case. For example, in the library setting, types of Intellectual Entities may include work, expression, manifestation or item to capture useful FRBR distinctions [Functional Requirements for Bibliographic Records [9]]. In an archival setting, Intellectual Entity types such as fonds and series are relevant and may be supported for the repository. Most repositories support discovery and delivery of Intellectual Entities such as a book, moving image, or article. But repositories may also choose to manage Intellectual Entities at a larger scale, such as a collection, or as a fine-grained component, such as an embedded table or image. It is up to the repository which types of Intellectual Entities it supports and represents in PREMIS. PREMIS supports this modeling through flexible relationship subtypes for capturing their relationships to other objects.

PREMIS users can choose to continue using Intellectual Entities to simply identify PREMIS-external descriptive metadata. If a repository does not manage Intellectual Entities, it does not need to record metadata about them.

8 Metadata Encoding & Transmission Standard (METS), http://www.loc.gov/standards/mets/.
9 IFLA, Functional Requirements for Bibliographic Records (Munich: K.G. Saur, 1998), http://www.ifla.org/VII/s13/frbr/frbr.pdf.

Representations

The goal of many preservation repositories is to maintain usable forms of Intellectual Entitiesover time. For an Intellectual Entity to be displayed, played, or otherwise made useable to a human, all of the files making up at least one form of that Intellectual Entity must be identified, stored, and maintained so that they can be assembled and rendered for a user at any given time. A Representation is the set of Files required to do this.

PREMIS chose the term “Representation” to avoid the term “manifestation” as it is used in the FRBR. In FRBR a manifestation entity is “all the physical objects that bear the same characteristics in respect to both intellectual content and physical form.”[10] In the PREMIS model a Representation is **a single instance** of *an Intellectual Entity held in a preservation repository*; note the difference in multiplicity ‘(all’ versus ‘a single instance’).

A preservation repository might hold more than one representation of the same Intellectual Entity. For example, the repository might acquire a single image (say, “Statue of a horse”) as a TIFF File. At some point the repository creates a derivative PEG2000 file from the TIFF and keeps both files. Each of these files would constitute a representation of “Statue of a horse.”

In a more complicated example, “Statue of a horse” might be a part of an article consisting of that TIFF image and a file of SGML-encoded text. If the repository created a JPEG2000 version of the TIFF, it would hold two Representations of the article: the TIFF and the SGML fileswould make up one representation, while the JPEG2000 and the SGML files would make up another representation. How those representations are stored is implementation specific. A repository might choose to store a single copy of the SGML file, which would then be shared between representations. Alternatively, the repository could choose to duplicate the SGML file and store two identical copies of it.

Not all preservation repositories will be concerned with representations. A repository might, for example, preserve File objects only and rely on external Agents to assemble these objects into usable representations. If he repository does not manage representations, it does not need to record metadata about them.

10 Ibid., p.21.

Files, Bitstreams, and Filestreams

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

A Bitstream as defined in the PREMIS data model is a set of bits embedded within a file. This differs from common usage, where a bitstream could in theory span more than one file. A good example of a file with embedded bitstreams is a TIFF file containing two images.

According to the TIFF file format specification a TIFF file must contain a header containing some information about the file. It may then contain one or more images. In the PREMIS data model each of these images is a bitstream and can have properties such as identifiers, location, inhibitors, and detailed technical metadata (e.g., color space).

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

The PREMIS data model refines the definition of **Bitstream** to include only those embedded bitstreams that cannot be transformed into standalone files without the addition of file structure (e.g., headers) or other reformatting to comply with some particular file format specification. Examples of these bitstreams include an image within a TIFF 6.0 file, audio data within a WAVE file, or graphics within a Microsoft Word file.

Some embedded bitstreams can be transformed into standalone files without adding any additional information, although a transformation process such as decompression, decryption, or decoding may have to be performed on the bitstream in the extraction process. Examples of these bitstreams include a TIFF within a tar file, or an encoded EPS within an XML file. In the PREMIS data model these bitstreams are defined as “filestreams”, that is, true files embedded within larger files.

Filestreams have all of the properties of **Files**, while **Bitstreams** do not. In the Data Dictionary, the column for “File” applies to both files and filestreams. The column for “Bitstream” applies to the subset of bitstreams that are not filestreams and that adhere to the stricter PREMIS definition of **Bitstream**. The location (*contentLocation* in the Data Dictionary) of a file would normally be a location in storage; while the location of a filestream or bitstream would normally be the starting offset within the embedding file.

Complete Examples

The relationship between Object categories can be illustrated by a couple of examples:

Example 1, *Animal Antics* (Figure 3): The book *Animal Antics* was published in 1902. A library digitized *Animal Antics*, creating one TIFF file for each of 189 pages. As structural metadata, it created an XML file showing how the images are assembled into a complete book. The library then performed OCR on the TIFF images, ultimately creating a single large text file that was marked up by hand in SGML. The library submitted 189 TIFF files, one XML file, and one SGML file to a preservation repository.

To the repository *Animal Antics* is an Intellectual Entity: it is a reasonable unit that can be described as a whole, with properties such as an author, a title, and a publication date. *Animal Antics* also has significant characteristics that need to be maintained through preservation actions such as the formatting and color of text. The repository has two representations, one consisting of 189 TIFF files and an XML file, and the other consisting of one SGML file. Each representation could render a complete version of *Animal Antics*, albeit with different unctionalities. The repository will record metadata about one Intellectual Entity, two Representation objects and 191 File objects.

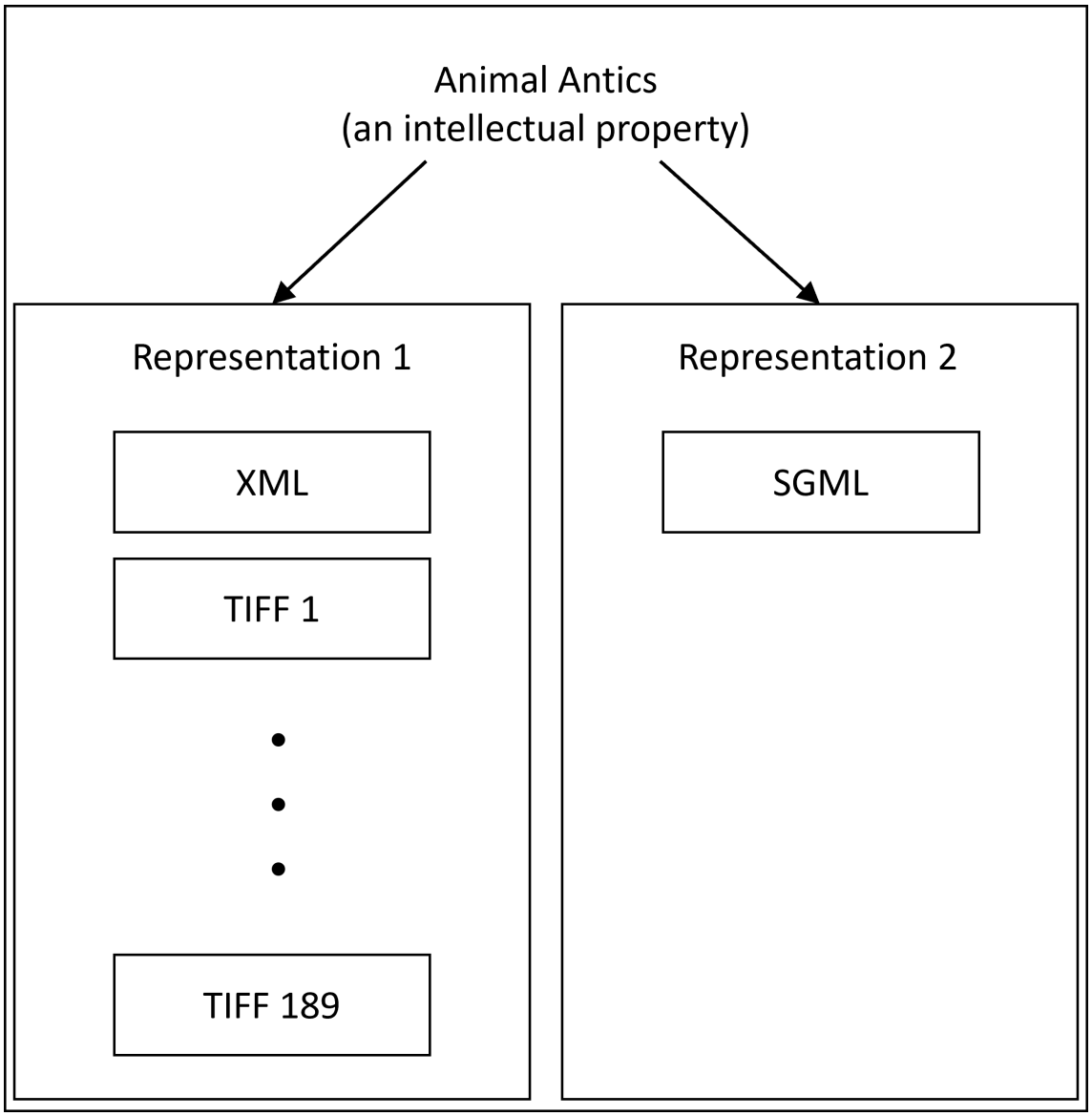

Figure 3. Animal Antics Intellectual Entity Example

Example 2, *Welcome to U* : Welcome to U, submitted to a preservation repository as an AVI (Audio Video Interleaved) File, is a 10-minute movie introducing new students to a university campus.

Welcome to U is an Intellectual Entity. The repository has one representation, which consists of a single AVI file. The repository's preservation strategy requires that it manage the audio bits of the AVI file separately from the video bits. The repository will record metadata about one Intellectual Entity object, one Representation object, one File object, and two Bitstream objects.

Environments

In order to preserve Digital Objects, repositories need to have information about the elements of the technical stack of software, hardware and other dependencies needed to correctly interpret the representations, files and bitstreams. Each element of such a technical stack is an environment.

Starting with PREMIS version 3.0, Object entities can be used to capture environments that are relevant to the content of the repository. Like content Objects, environments can be described at all Object category levels (Intellectual Entity, Representation, File, Bitstream). Intellectual Entities are used to record descriptive information for an environment. An environmentIntellectual Entity can be used to define its function (e.g., is it a software application or a hardware peripheral) and designation (including name and version information). It may reference one or more corresponding entities in an external registry, such as PRONOM or UDFR.

The repository may or may not hold an actual implementation of such an environment (e.g., hold executable software) as a Representation, File and/or Bitstream. These can provide access to content bitstreams, files or representations held in the repository. These actual implementations can be an identical or functionally equivalent technical stack to the stack used originally. For example, if a repository holds video files it might, in addition, hold software capable of rendering such videos and an image of an operating system capable of running that software.

A software Agent modeled in PREMIS can be related to the environment Intellectual Entities, Representations, Files or Bitstreams that describe and capture the Agent.

Environments are discussed in further detail in the section “Special Topics”

More on Events

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

It is up to the repository which actions to record as events. Some actions may be considered too trivial to record, or may be recorded in other systems (as, for example, routine file backups may be recorded in storage management systems). It is also an implementation decision whether to record events that occur before an object is ingested into the preservation repository, for example, derivation from an earlier object, or changes of custody. In theory, events following the deaccessioning of an Intellectual Entity could also be recorded. For example, a repository might first deaccession an Intellectual Entity, then delete all file objects associated with that entity, and record each deletion as an event.

In the data model Objects are associated with events in two ways. If an object is related to a second object through (because of) an event, the event identifier is recorded in the *relationship* container as the semantic component *relatedEventIdentifier*. If the object simply has an associated event with no relationship to a second object, the event identifier is recorded in the container *linkingEventIdentifier*. (For more information on relationships, see section "Relationships between Objects".)

For example, assume a preservation repository ingests an XML file (object A) and creates a normalized version of it (object B) by running a program (event 1). In the metadata for object B, this could be recorded in *relationship* as follows:

relationshipType = “derivation”
relationshipSubType = “has source”
relatedObjectIdentifier
  relatedObjectIdentifierType = “local”
  relatedObjectIdentifierValue = “A”
relatedEventIdentifier
  relatedEventIdentifierType = “local”
  relatedEventIdentifierValue = “1”

Note that since sequencing is not applicable, the semantic unit *relatedEventSequence* is not included.

Continuing with this example, assume that after object B is created it is validated by running another program (event 2). In this case event 2 pertains only to object B, not to the relationship between B and A. The link to event 2 would be recorded as *linkingEventIdentifier*:

linkingEventIdentifierType = “local”
linkingEventIdentifierValue = “2”

A given object can be associated in these two ways with any number of events.

All events have outcomes (success, failure, etc.). Some events also have outputs; for example, the execution of a program creates a new file object. The semantic units *eventOutcome* and *eventOutcomeDetail* are intended for documenting qualitative outcomes. For example, if the event is an act of format validation, the value of *eventOutcome* might be a code indicating the object is fully valid. Alternatively, it might be a code indicating the object is not fully valid, and eventOutcomeDetail could be used to describe all anomalies found. If the program performing the validation writes a log of warnings and error messages, a second instance of *eventOutcomeDetail* could be used to store or point to that log.

If an event creates objects that are stored in the repository, those objects should be described as entities with a complete set of applicable metadata and associated with the event by links. Some additional aspects of an event other than its outcomes or outputs might be recorded, such as the specific parameters used during a migration event, the nature of the operation (automated, manual or semi-automated) and so on. Such information can be recorded in *eventDetail*.

More on Agents

Agents are clearly important but are not the focus of the Data Dictionary, which defines only a means to identify the agent and a classification of agent type (person, organization, or software). While more metadata is likely to be necessary, this is left to other initiatives to define in detail. In version 3.0, the *agentVersion* semantic unit is added to the Data Dictionary.

The data model diagram shows how Agents relate to other entities. An Agent can be related to Rights Statements in which the agent has an interest. An Agent can also relate to the Event entity in which the Agent takes an action. Each Event can have one or more related Agents. Because a single Agent can perform different roles in different Events, the role of the Agent is a property of the Event entity, not of the Agent entity. Agents relate to Objects in 2 ways: 1) Agents affect Objects that are involved in an Event; 2) Agents are software Agents that are described through an environment Object. In the first case Agents influence Objects only indirectly through Eventsand are not directly linked to the Object. In the second case the environment Object further describes and captures the software Agent. For this situation there is a direct link (*linkingEnvironmentIdentifier*) from the Agent to the environment.

More on Rights

Many efforts are concerned with metadata related to intellectual property rights and permissions, from rights expression languages to the <indecs> [11] framework. However, only a small body of work addresses rights and permissions specifically related to digital preservation. After the publication of the first edition of the PREMIS Data Dictionary, the Library of Congress in its capacity as PREMIS Maintenance Agency commissioned a paper, "Rights in the PREMIS Data Model," by Karen Coyle[12]. This paper discussed copyright, licenses, and statute as three bases for establishing intellectual property rights, and recommended an expansion of the rights information in the Data Dictionary to include information on these bases.

Consequently, the *permissionStatement* in the original Data Dictionary was replaced with the *rightsStatement* in version 2.0. In that revision the Editorial Committee relied heavily upon the Coyle paper, background materials such as Peter Hirtle's "Digital Preservation and Copyright,[13]" and he California Digital Library's draft copyrightMD schema[14]. It should be noted that the proposed uses of copyrightMD and PREMIS Rights are rather different. The copyrightMD schema is intended to document factual information to allow a human being to make an informed copyright assessment of a given work. The PREMIS *rightsStatement* is intended to allow a preservation repository to determine whether it has the right to perform a certain action in an automated fashion, with some documentation of the basis for the assertion.

Version 2.2 added semantic units to *rightsStatement* to specify a rights basis other than copyright, license or statute (e.g. institutional policy); to be able to link to further information about the rights through a documentation identifier; to associate applicable dates with a Rightsstatement; and to allow for term of restriction in addition to the existing term of grant.

Version 3 contains no new additions or changes related to Rights Statements.

11 Godfrey Rust and Mark Bide, The <indecs> Metadata Framework: Principles, Model, and Data Dictionary, June 2000, http://www.doi.org/topics/indecs/indecs_framework_2000.pdf

12 Coyle, Karen, Rights in the PREMIS Data Model, http://www.loc.gov/standards/premis/Rights-in-the-PREMIS-Data-Model.pdf.

13 Hirtle, Peter B., Digital Preservation and Copyright, http://fairuse.stanford.edu/commentary_and_analysis/2003_11_hirtle.html.

14 California Digital Library, copyrightMD schema, http://www.cdlib.org/inside/projects/rights/schema/.

General Topics on the Structure and Use of the Data Dictionary

The semantic units defined in the PREMIS Data Dictionary are bound together by a few structural conventions that help organize the Data Dictionary and support its implementation. These conventions include the use of identifiers; the manner in which relationships are handled in the Data Dictionary; and the "1:1 Principle" relating metadata to Objects.

Identifiers

Instances of Objects, Events, Agents, and Rights statements are uniquely identified by a set of semantic units collected under "Identifier" containers. These semantic units follow an identical syntax and structure, regardless of entity type:

[entity type]Identifier
  [entity type]IdentifierType: domain in which the identifier is unique
  [entity type]IdentifierValue: identifier string

The following examples illustrate the use of this syntax to identify an Object residing in Harvard's Digital Repository Service (DRS), and an event that occurs under the auspices of the NRS (Name Resolution Service):

Example 1: Identifying an Object
ObjectIdentifier
  ObjectIdentifierType: NRS
  ObjectIdentifierValue: http://nrs.harvard.edu/urn-3:FHCL.Loeb:sa1

Example 2: Identifying an Event
EventIdentifier
   EventIdentifierType: NRS
  EventIdentifierValue: 716593

In both examples, the identifier type is "NRS", which indicates that the identifier is unique within the domain of the Name Resolution Service that assigns identifiers for the Digital Repository Service. Identifier type should be defined as specifically as possible, and provide sufficient information to indicate the relevant naming authority, as well as how to build the identifier value. For example, it would have been permissible to use "URL" for *objectIdentifierType* in the first example, since the identifier value is unique in that domain, but "NRS" conveys more information about the domain in which the identifier is created and used.

If all identifiers are local to the repository system, it is unlikely that the identifier type would need to be explicitly recorded for each identifier in the system. This is an example of a semantic unit whose information is known implicitly by context or policy, and is therefore not implemented as a metadata element in the preservation system. However, if the repository exchanges digital objects and their associated metadata with other repositories, the identifier type should be supplied explicitly.

Identifiers can be created internally or externally to the repository. The PREMIS Data Dictionary does not require or even recommend a specific identifier scheme; this is an implementation-specific issue and is therefore outside the scope of the Data Dictionary. The Data Dictionary simply provides a general syntax that can be used to express identifier type and value, regardless of the

specific scheme chosen. It is recommended, however, that repositories choose persistent identification schemes wherever possible.

Identifiers are repeatable for Objects and Agents; they are not repeatable for Rights and Events. Objects and Agents often have multiple identities in a global environment, and across systems, and therefore are likely to have multiple identifiers. Rights and Events are considered to have a context limited to a particular preservation repository, and therefore do not require multiple identifiers.

Identifiers are used as references to establish relationships between entities in the PREMIS data model. Relationships are discussed in the next section.

Relationships between Objects

As noted earlier, an Object in a repository can be related to one or more other Objects in the repository. The PREMIS Data Dictionary supplies semantic units to support documentation of relationships between Objects. A wide range of metadata facts are expressed as relationships—for example, "is migrated from," "is keyed text of," "is thumbnail of." In some cases these relationship statements combine more than one fact (e.g., "is keyed text of" combines "is a keyed text" and "is derived from") and many existing metadata frameworks specify relationship types, such as the Dublin Core Relation element (IsPartOf, IsFormatOf, IsVersionOf, etc. [15]). Most relationships among content Objects appear to be variants of two basic types: structural and derivation relationships. Dependency relationships are primarily used to help express relationships between environment Objects.

Structural relationships show relationships between parts of Objects. The structural relationships between the files that constitute a representation of an Intellectual Entity are clearly essential preservation metadata. If a preservation repository can't put the pieces of a digital object back together, it hasn't preserved the object. For a simple digital object (e.g., aphotograph) structural information is minimal: the file constitutes the representation. Other digital objects such as e-books and web sites can have quite complex structural relationships.

Derivation relationships result from the replication or transformation of an Object. The intellectual content of the resulting Object is the same, but the Object's instantiation, and possibly its format, are different. When file A of format X is migrated to create file B of format Y, a derivation relationship exists between A and B.

Many digital objects are complex, and both structural and derivation information can change over time as a result of preservation activities. For example, a digitized book represented by 400 TIFF page images might after migration become four PDF files each containing 100 pages.

A structural relationship among objects can be established by an act of derivation before the objects were ingested by the repository. For example, a word-processing document could have been used to create derivative files in PDF and XML formats. If only the PDF and XML files are submitted to the preservation repository, these objects are different representations of the same Intellectual Entity with parent-child relationships to the source word-processing file. They do not have derivation relationships with each other, but do have a structural relationship as siblings (children of a common parent).

There is no one way to model all possible structural or derivation information. Essential information that must be captured is described in the semantic

15 The Dublin Core Metadata Element Set, DCMI Terms, http://www.dublincore.org/documents/dcmi-terms.

components of the semantic unit *relationship*. Structural and derivative relationships link Objects; the Objects must be identified, the type of relationship must be identified in some way (e.g., “is child of”) and the relationship may be associated with an event that created that relationship. Implementers will likely choose approaches that best suit the content to be preserved by using, for example, the METS structMap or descriptive metadata schemes that define relationship types (e.g. Dublin Core[16]).

A more detailed set of relationships need to be expressed when environment Objects are involved. For example, a **dependency relationship** exists when an Object requires an environment Object to support its function, delivery, the coherence of its content, or it may require specific software or hardware. Similarly, an environment Object may depend on other environment Objects. Figure 4 depicts a dependency relationship between a .txt content Object and a Libre Office software environment Object. It also shows a dependency between the Libre Office software environment Object and a Java Runtime environment Object. In this way requirements for hardware and software are brought together with requirements for dependent files to form a complete picture of the information or assets required for the rendering and/or understanding the Object. Suggested types of relationships involving environment Objects are **structural**, **derivation**, **replacement**, **dependency**, and **reference**. Suggested subtypes are:

- includes
- is included in (structural)
- represents
- is represented as (structural)
- supersedes (replacement)
- is superseded by
- requires
- is derived from
- is source of
- is required by (dependency)
- is deployed on (dependency)
- documents
- is documented in

16 The Dublin Core Metadata Element Set, http://www.dublincore.org/documents/dces/.

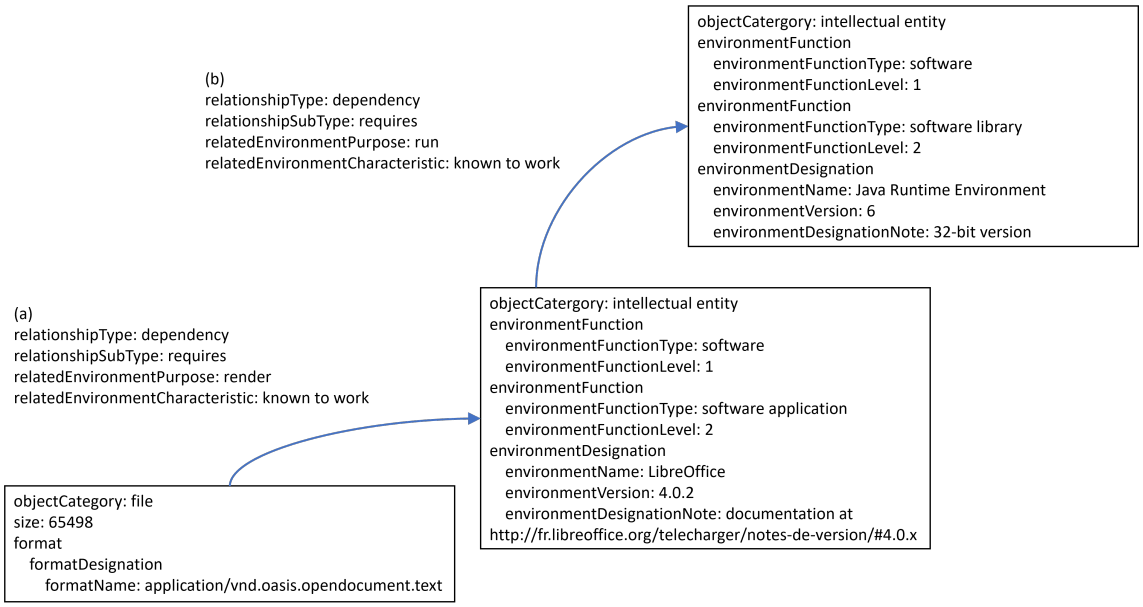

Figure 4. Dependency relationships between (a) a content and environment Object and (b) between two environment Objects.

Relationships between Entities of different types

The data model diagram uses arrows to show relationships between entities of different types. Objects are related to Events, Agents are related to Events, etc. The Data Dictionary expresses relationships as linking information by including in the information for entity A, a pointer to the related entity B. Every entity in the data model has a unique identifier, which can be used in a pointer. So, for example, the Object entity has arrows pointing to Rights and Events. These are implemented in the Data Dictionary by the semantic units *linkingRightsStatementIdentifier* and *linkingEventIdentifier*.

The 1:1 principle

In digital preservation it is common practice to create new copies or representations of objects. For example, in format migration file A in format X may be input to a program which outputs file B in format Y. There are two ways to think about files A and B. One might think of them as a single Object, the history of which includes the transformation from X to Y, or one could think of them as two distinct Objects with a relationship created by the transformation event.

The 1:1 principle in metadata asserts that each description describes one and only one resource. As applied to PREMIS metadata, files, bitstreams and representations held within the preservation repository are described as a static set of bits. It is not possible to change a file (or bitstream or representation); one can only create a new file (or bitstream or representation) that is related to the source Object. In the example above, therefore, files A and B are distinct Objects with a derivative relationship between them. Similarly, the 1:1 rule applies to Intellectual Entities, in that an Intellectual Entity should not be reinterpreted to stand for a different entity. While it is possible to change the description of an Intellectual Entity without changing its identity, a new Intellectual Entity must be created if a change would result in a different intellectual interpretation. For example, the identity of an Intellectual Entity is not changed if new structural links to other Intellectual Entities are created or if an additional physical representation is created for it. The identity of an Intellectual Entity would be changed if the unit of content comprising the Intellectual Entity were to change significantly; for example, the coverage of a collection, or different revisions of an article that do not have the same status nor intellectual content and should result in different Intellectual Entities.

The Data Dictionary has a semantic unit for the creation date of an Object (*dateCreatedByApplication*) but not for the modification date of an Object, because an Object, by definition, cannot be modified.

When new objects are derived from existing objects the event that created the new object should be recorded as an Event, which will have a date/time stamp. The relationship(s) among the objects should be recorded using the relationshipsemantic unit associated with the Object entity. The semantic component *relatedEventIdentifier* should be used to make the association with the Event.

Implementation Considerations

PREMIS conformance

The PREMIS Data Dictionary was designed to be as flexible as possible in its implementation. No assumptions were made regarding the nature of the digital preservation system in which the Data Dictionary would be implemented, the preservation strategy being followed, or even the metadata management processes responsible for creating and maintaining preservation metadata. The "technical neutrality" built into the design of the Data Dictionary is intended to maximize the Dictionary's applicability across the broad range of digital preservation contexts in which it could potentially be implemented.

Technical neutrality does not, however, override the need to establish a set of principles for implementing the Data Dictionary in ways that ensure data consistency within and across preservation repositories. Such consistency is necessary in order to support a variety of use cases, including:

- Inter-repository data exchange
- Repository certification
- Shared registries
- Automation/reusable tools
- Vendor support

To support these and other use cases, the PREMIS Editorial Committee released a Conformance Statement in 2010[17] This defined a set of principles "governing a conformant implementation" of the Data Dictionary. A key tenet of the work was to give implementers of the Data Dictionary flexibility to be able to use the Dictionary in ways that allowed them to respond to their own internal preservation processes.

Because experience of digital preservation practices and implementing PREMIS has grown across the community since the original Conformance Statement was drafted, in 2014 the Editorial Committee asked a sub-committee to explore again the notion of conformance. This resulted in a new Conformance Statement, issued in April 2015, which builds on the original Statement while introducing new principles designed to assist implementers to quantify the degree to which PREMIS has been implemented by a repository.[18] The new Statement accomplishes this by introducing graduated levels of conformance, which draw a distinction between metadata that can be mapped to the PREMIS Data Dictionary, metadata which can be exported as PREMIS, and metadata which are natively conformant to the Data Dictionary without any further mapping or conversion. The levels of conformance are further refined by whether a repository implements the Object entity only (considered a minimum requirement for any assertion of conformance) or whether the Events and Agents entities are also implemented.

17 PREMIS Editorial Committee, Conformance Implementation of the PREMIS Data Dictionary, October2010, http://www.loc.gov/standards/premis/premis-conformance-oct2010.pdf.

18 http://www.loc.gov/standards/premis/premis-conformance-20150429.pdf

Unlike the original Statement, the new version does not explicitly address the concepts of internal and external conformance (i.e. internal use within a repository versus interchange with other repositories); however, as a repository progresses to levels two and three of conformance, the greater its ability to exchange preservation metadata with other repositories. It is important to note that adherence to the conformance principles is not a formal requirement for implementing the PREMIS Data Dictionary (although the Editorial Committee does believe that following these principles would be good practice in nearly all implementation contexts). In other words, a repository is free to implement the Data Dictionary in whatever way it chooses in situations where conformance is not asserted. However, in situations where PREMIS conformance is asserted, implementers must be able to demonstrate the level with which they purport to comply.

The revised Statement adds new content on the subject of controlled vocabularies, addressing in particular the vocabulary for the *eventType* semantic unit. The Statement requires that, for higher levels of conformance, a repository must record "sufficient Event metadata to document actions the repository has taken to preserve the digital objects." The PREMIS Editorial Committee recognizes that it cannot prescribe what actions a repository must take to preserve digital objects. However, the case can be made that using consistent terms for those actions is essential to building a body of understanding of what constitutes core preservation functions and how to describe them, beyond the basic, high-level functional areas (ingest, data management, archival storage, administration, access and preservation planning) defined in the OAIS functional model. The Conformance Statement therefore makes an explicit connection between PREMIS conformance, the identification of core preservation functions and the use of accepted terms to describe those functions.

The revised conformance statement is available at: http://www.loc.gov/standards/premis/premis-conformance-20150429.pdf.

Implementation of the data model

The PREMIS data model is meant to clarify the meaning and use of the semantic units in the Data Dictionary. It is not intended to prescribe an architecture for implementation.

The working group believed that most preservation repositories will need to deal in some way with the conceptual entities Objects, Agents, Events, and Rights, and found it useful to distinguish between the properties of categories of Objects, such as Intellectual Entities, Representations, Files and Bitstreams. A particular repository implementation, however, may need to be more or less granular or define different categories of entity altogether. PREMIS recommends that any data model used be clearly defined and documented, and that metadata decisions be consistent with the data model.

Sets of semantic units may be grouped and related indirectly to particular entities. For example, Objects may be linked to their environment Objects using the semantic container "*relationship*" with *relationshipType*="dependency" and *relationshipSubType*="requires" specifying the *relatedObjectIdentifier* of the environment. Logically, each file has one or more associated environments. This could be handled in many different ways by different implementations. For example:

- Repository 1 uses a relational database system. It has a "file" table with a row for each file object, and an "environment" table with a row for each unique environment object. The "file" table can be joined with the "environment" table to get the appropriate environment information for each file.
- Repository 2 models representations as containers and files as objects within those containers. Each Object consists of a set of property/typed value pairs. Properties define roles for values. Property and type descriptions are themselves objects whose identifiers are drawn from the same namespace as other object identifiers. A file object may include an environment property which in turn would point to an environment object. This property could be a shortcut for the PREMIS defined relationship link to required environments.

In many cases the environment is determined by the file format; that is, all files of a particular format will have the same environment information.

- Repository 3 uses an externally-maintained registry to obtain environment information. It maintains an internal inventory of file formats and their access keys for the external registry. Environment information is accessed via a web services interface to the external registry and obtained dynamically when needed.

Storing metadata

Most commonly, metadata is stored in relational database tables, as XML documents in an XML database, as RDF datasets in an RDF triple store, or as XML documents stored with the content data files. Other methods include proprietary flat file formats and object-oriented databases. Many repositories use two or more of these methods. Storing metadata elements in a database system has the advantages of fast access, easy update, and ease of use for query and reporting. Storing metadata records as digital objects in repository storage along with the digital objects the metadata describes also has advantages: it is harder to separate the metadata from the content, and the same preservation strategies that are applied to the content can be applied to the metadata. Recommended practice is to store critical metadata in both ways.

Compound objects require structural metadata to describe the internal structure of the objects and the relationships between their parts. In the PREMIS Data Dictionary, semantic units that begin with “related” and “linking” can be used to express certain simple structural information. In some cases this will be adequate for the use of the Object, and in other cases it will not be. Often the presentation, navigation and/or processing of an Object will require rich structural metadata recorded according to some other standard, such as METS, MPEG-21[19] or SMIL[20]. In this case the file containing the structural metadata would be a File Object to be preserved in its own right. Regardless of whether a file of independent structural metadata exists as part of the representation, when an archived representation is exported to another repository, the metadata linking files and representations should be provided.

19 Information technology – Multimedia framework (MPEG-21), ISO/IEC 21000 (multiple parts), International Organization for Standardization.

20 Synchronized Multimedia Integration Language (SMIL), http://www.w3.org/TR/REC-smil/.

Supplying metadata values

Most preservation repositories will deal with large quantities of materials, so it is desirable to automate the creation and use of metadata as much as possible. The values of many PREMIS semantic units can be obtained by parsing files programmatically, or can be supplied as constants by repository ingest programs. In cases where human intervention might be unavoidable, PREMIS tends to pair a semantic unit requiring a coded value with a second semantic unit allowing a textual explanation.

When information is supplied by the individual or organization submitting the objects to the repository, recommended practice is for the repository to attempt to verify this information by program whenever possible. For example, if a file name includes a file extension, the repository should not assume the file extension necessarily indicates the format and should attempt to verify the format of the file before recording this as metadata.

To facilitate automatic processing, the use of controlled vocabularies is recommended for a number of PREMIS semantic units. PREMIS assumes that repositories will adopt or define controlled vocabularies useful to them. The Data Dictionary indicates where best practice would require use of a controlled vocabulary. It does not require specific controlled vocabularies, although in the current version it shows some examples of usage (In previous versions it indicated suggested values.)

The PREMIS Editorial Committee concluded that implementers should be able to choose the vocabulary and specify which vocabulary is used. Whether and how to validate that the appropriate values have been used is an implementation consideration. Since version 2.0 of the PREMIS Data Dictionary, the PREMIS Maintenance Activity at the Library of Congress has established a web service for lists of controlled values to be used with many of the PREMIS semantic units that recommend use of a controlled vocabulary and were considered lists that could be broadly applicable (as opposed to those that were by definition local vocabularies). These are available at the Library of Congress Linked Data Service for Authorities and Vocabularies (http://id.loc.gov/vocabulary/preservation). New terms and new vocabularies will be added as they are requested. Repositories may use these or define their own, but it should be clear what the source of each controlled vocabulary is when exporting metadata for exchange. Interoperability is enhanced if common vocabularies are used and declared.

An implementer may choose to document controlled vocabularies used in its repository so that exchange partners will know what to expect as values in the metadata. For instance, METS users may specify controlled vocabularies used in metadata in a METS profile, or PREMIS profiles may be established to document the same. In the PREMIS XML schema version 2.3 a mechanism was added for the convenience of users to allow for the specification of the vocabulary used, either in the form of a string or a URI. Values may also be

expressed as a URI. The following information may be given in relation to a PREMIS semantic unit that uses a controlled vocabulary and is documented in the PREMIS XML schema:

- Authority: The controlled vocabulary from which the value is taken, in string form.
- Authority URI: The controlled vocabulary from which the value is taken, identified by a unique URI.
- Value URI: The value from the controlled vocabulary, identified by a unique URI.

Where controlled vocabularies are recommended under a data constraint, these attributes may be used to indicate the vocabulary, and the value URI may indicate the controlled term in the form of a URI. In this version of the Data Dictionary a specific source vocabulary from http://id.loc.gov/vocabulary/preservation is referenced in each semantic unit for which this is applicable, but implementers may choose other vocabularies or implementations.

In Resource Description Framework (RDF), use of resource URIs as property values is encouraged[21], and many XML Schemas require attribute values to be URIs.[22]

In general, resource URIs are allowable as values for semantic units in the PREMIS Data Dictionary, unless some noted constraint would disallow this (e.g. date or integer datatype constraint). Whatever the technology used to identify, document and maintain values within the repository, the repository should be able to supply the controlled values in some way. Therefore, most examples in the Data Dictionary are plain text values rather than resource URIs. For example, the equivalent of a “migration” value for an eventType semantic unit might be:

- “Migration,” which is an implementation-agnostic constant whose meaning and/or documentation is known to the repository through some table or other documentation under the control of the repository organization.
- A locally-defined and maintained URI (e.g. http://example.org/events/migration), or an external one whose maintainer is trusted by the repository (in the former case: http://id.loc.gov/vocabulary/preservation/eventType/mig).

Which of the above to choose is an implementation decision.[23]

21 Resource Description Framework (RDF), http://www.w3.org/RDF/.

22 For example, in the XML-Signature Syntax and Processing(XMLDsig), the value of the signature method algorithm must be a URI, such as http://www.w3.org/2000/09/xmldsig#dsa-sha1

23 However, it is strongly recommended to use URIs for values when using the OWL ontology to express PREMIS.

Extensibility

For several semantic units the Data Dictionary notes the potential for extensibility, to allow implementations to include additional local metadata or to provide additional structure or granularity of metadata, if required. The inclusion of such additional metadata is relatively simple for implementations using relational databases; however, a mechanism for including such metadata when using the PREMIS schemas was not available in the first release of the Data Dictionary and schemas. Version 2.0 of the Data Dictionary introduced a formal mechanism for extensibility within the schemas for a small number of semantic units which were deemed prime candidates for extension. Version 3.0 has added the following extensible semantic units: eventDetailInformation and environmentDesignation. Later revisions of the Data Dictionary may add to this initial set of extensible semantic units if warranted.

The semantic units for which extensibility is supported in the schemas are:

- significantProperties [Object entity]
- objectCharacteristics [Object entity]
- creatingApplication [within objectCharacteristics, Object entity]
- environmentDesignation [within Environment, Object entity]
- environment [within Object entity]
- signatureInformation [Object entity]
- eventDetailInformation [Event entity]
- eventOutcomeDetail [within eventOutcomeInformation, Event entity]
- rights [Rights entity]
- agent [Agent entity]

These semantic units may be extended by use of an extension container within the Data Dictionary and schemas. Within the Data Dictionary, a corresponding semantic unit is indicated within the defined semantic components for each of the semantic units listed above as an extensible container with *extension* added to the name of the container that it extends. An extension may contain metadata encoded according to an external schema.

In devising the mechanism for extensibility, the PREMIS Editorial Committee adopted the rinciple that only semantic units which are containers may be extended. This would enable the use of a PREMIS defined semantic unit and/or a container for semantic units defined outside of PREMIS. This required some structural change (i.e. the addition of a container). For example the semantic unit *eventDetail* underwent such change with the creation of a container *eventOutcomeDetail*.

In utilizing the extensibility mechanism with the listed extensible semantic units, the following principles should be observed:

- An **extension** container may be used to supplement PREMIS semantic units with additional metadata or replace PREMIS semantic units with other applicable metadata within the parent container. The one

exception is *objectCharacteristicsExtension*, which may only supplement *objectCharacteristics*.

- Where there is a one-to-one mapping between the contents of an **extension** container and an existing PREMIS semantic unit, recommended best practice would be to use the PREMIS semantic unit rather than its equivalent in the extension; however, implementers may choose to use the extension alone, if circumstances warrant.
- If any semantic unit is not used it should be omitted, rather than an empty schema element included.
- If the information in an **extension** container needs to be associated explicitly with a PREMIS subunit, the parent container should be repeated and contain the appropriate subunit and the extension container. Also, if extensions from different external schemas are needed, the parent container should be repeated. In this case the repeated parent container may include the extension container with or without any other existing PREMIS semantic units for that parent container.
- When an **extension** container is used, the external schema being used within that extension container must be declared.

Additional information may be given about the extension metadata and is provided for in the PREMIS XML schema. This includes:

- Date the metadata was created
- Status of the metadata
- Internal IDs to provide links
- Type of metadata (i.e. the metadata scheme) and version
- Message digest and message digest algorithm of the metadata
- Type of location identifier when reference is to external metadata

Please note that extension containers were dropped in the PREMIS OWL ontology, as combining different vocabularies in the same description is a built-in capability of RDF.

Date and time formats in PREMIS

All semantic units that specify the use of a date or a date and a time suggest the use of a structured form to aid machine processing. In keeping with its being implementation independent, the Data Dictionary does not specify a particular standard to be used. In some cases, conventions are needed to express other aspects of a time period, such as an open-ended or questionable date. The PREMIS XML schema specifies date and time formats and establishes such conventions; it is recommended that these be used when needed. The following are semantic units that may include a date or date and time:

- preservationLevelDateAssigned (under preservationLevel)
- dateCreatedByApplication (under creatingApplication)
- eventDateTime (under Event)
- copyrightStatusDeterminationDate (under copyrightInformation)
- statuteInformationDeterminationDate (under statuteInformation)
- startDate (under statuteApplicableDates, copyrightApplicableDates, otherRightsApplicableDates, termOfGrant, and termOfRestriction)
- endDate (under statuteApplicableDates, copyrightApplicableDates, otherRightsApplicableDates, termOfGrant, and termOfRestriction)

THE PREMIS DATA DICTIONARY VERSION 3.0

The PREMIS Data Dictionary includes semantic units for Objects, Events, Agents, and Rights. The template for each entry includes a place for notes about how to create or use the semantic unit. In some cases the group felt additional information, such as the reason for a semantic unit's definition or issues that arose in the group's deliberations, would be useful; for these details, see section "Special Topics".

A semantic component always inherits the applicability of the containing semantic unit. That is, if the containing semantic unit specifies that it is applicable to files but not to representations, each of its semantic components is applicable to files and not to representations. Repeatability and obligation, however, may vary.

Each entry in the Data Dictionary offers these attributes of a semantic unit:

Name of the semantic unit: Names were devised to be descriptive and unique within the Data Dictionary. Using these names for the exchange of metadata among preservation repositories will aid interoperability. These names need not be used internally within any individual preservation repository.

Semantic components: The semantic components each have their own entries later in the Data Dictionary. A semantic unit that has semantic components does not have any value of its own. Only semantic units at the lowest level have values.

Definition: The meaning of the semantic unit.

Rationale: Why the semantic unit is needed, if this is not self-evident from the definition.

Data constraint: How the value of the semantic unit should be encoded. Some common data constraints are:

> *Container* – The semantic unit is an umbrella for two or more semantic components and has no value of its own.
> *None* – The semantic unit can take any form of value.
> *Value should be taken from a controlled vocabulary* – The preservation repository should establish an authority list of values that are useful and meaningful to the repository. The PREMIS Data Dictionary does not specify what this authority list should be, and it is assumed that different repositories will use different vocabularies. In general, when a value is taken from a controlled vocabulary, the source of the vocabulary should be recorded. A mechanism to record the source is provided in the PREMIS XML schemas.

Object category: Whether the unit applies to an Intellectual Entity, Representation, File, or Bitstream Object. Semantic units that apply to

Representations also apply to Intellectual Entities and semantic units that apply to Files also apply to Bitstreams (see section "More on Objects").

Applicability: A scope of “applicable” means it applies to that category of Object.

Examples: One or more examples of values the semantic unit may take. Examples are intended to be illustrative.

An example of an actual value is set in normal text. Text in brackets presents a description of the value rather than the value itself. For example, “SHA-1 message digest” reflects the actual value of the semantic unit, while “[SHA-1 message digest]” means the value of the semantic unit is a SHA-1 message digest such as:

“7c9b35da4f2ebd436f1cf88e5a39b3a257edf4a22be3c955ac49da2e2107b67a1924419563”

Repeatability: A semantic unit designated as “Repeatable” can take multiple values. It does not mean that a repository must record multiple instances of the semantic unit.

Obligation: Whether a value for the semantic unit is mandatory (if applicable) or optional.

A mandatory semantic unit is something that the preservation repository needs to know, independent of how or whether the repository records it. The repository might not explicitly record a value for the semantic unit if it is known by some other means (e.g., by the repository’s business rules). “Mandatory” actually means “mandatory if applicable.” For example, an identifier for a bitstream is mandatory only if the repository manages data at the bitstream level. When exchanging PREMIS-conformant metadata with another repository, values for mandatory semantic units must always be provided.

Values for optional semantic units are encouraged but not required.

If a container unit is optional, but a semantic component within that container is mandatory, the semantic component must be supplied if and only if the container unit exists. That is, if a value for any of the optional or mandatory semantic units in the container is supplied, a value for all of the mandatory semantic units in the container must be supplied.

Creation/Maintenance notes: Notes about how the values for the semantic unit may be obtained and/or updated.

Usage notes: Information about the intended use of the semantic unit, or clarification of the definition.

Limits to the scope of the Data Dictionary

Descriptive metadata: Typically, descriptive metadata is used to describe Intellectual Entities. Nearly all preservation repositories either include descriptive metadata or link to descriptive metadata located outside the repository itself. Such metadata may identify a resource by publication information such as creator and title, or may characterize its intellectual content through classification, subject terms, and so on. Descriptive metadata can be important both for discovery of archived resources and for helping decision makers during preservation planning. However, the Data Dictionary does not focus on descriptive elements for two reasons.

First, descriptive metadata is well served by existing standards. MARC[24], MODS[25], the Dublin Core Metadata Element Set, the Content Standard for Digital Geospatial Metadata[26], the VRA Core[27], the Encoded Archival Description (EAD)[28], and the Data Documentation Initiative[29] schemas are only some of the standards that define descriptive metadata elements. The working group did not want to add another set of descriptive elements to an already crowded field. Second, descriptive metadata is often domain specific. For the purposes of preservation it is less crucial that a common set of elements describe, for example, satellite telemetry and digital Picassos than that communities of interest be able to capture and exchange information in a form that reflects their materials and interests appropriately.

Agents: PREMIS does not define the characteristics of Agents in any detail. Metadata describing people, organizations, and other entities that can act as Agents has been defined in many existing formats and standards, such as MARC, vCard[30] MADS[31], and several other schemes currently under development. As long as a preservation repository can properly identify Agents that have acted upon Objects in its care, additional Agent characteristics will be determined by local requirements; many can be modeled on existing standard metadata element sets.

Rights: PREMIS primarily defines characteristics of Rights and permissions concerned with preservation activities, not those associated with access and/or distribution. The semantic units allow for extensibility to use an external Rights metadata scheme.

Technical metadata: Technical metadata describes the physical rather than intellectual characteristics of digital objects. Detailed, format-specific technical metadata is clearly necessary for implementing most preservation strategies,

24 MARC 21, http://www.loc.gov/marc/
25 Metadata Object Description Schema (MODS), http://www.loc.gov/standards/mods/.
26 Content Standard for Digital Geospatial Metadata, FGDC-STD-001-1998, http://www.fgdc.gov/metadata/csdgm/.
27 VRA Core 4.0, http://www.vraweb.org/projects/vracore4/.
28 Encoded Archival Description (EAD), http://www.loc.gov/ead/.
29 Data Documentation Initiative (DDI), http://www.ddialliance.org/.
30 vCard, http://www.imc.org/pdi/.
31 Metadata Authority Description Schema (MADS), http://www.loc.gov/standards/mads/.

but the group had neither the time nor the expertise to tackle format-specific technical metadata for various types of digital files. Therefore, it restricted the technical metadata included in the Data Dictionary to the semantic units it believed apply to objects in all formats. Further development of technical metadata is left to format experts. An extensibility mechanism is provided by including the semantic unit *objectCharacteristicsExtension*, which may be used with an external technical metadata scheme.

Media details: The working group did not attempt to define metadata for detailed documentation of media. For example, PREMIS defines a semantic unit for identifying the medium on which an object is stored. A preservation repository will probably want to know more detailed information about the media employed. If the repository stores data on DVDs, for example, it may need to know the specific technical characteristics of the specific DVD units, such as manufacturer, dye material, and dye thickness. PREMIS leaves the definition of metadata for describing media characteristics to specialists in these areas.

Business rules: The working group made no attempt to describe the business rules of a repository, although certainly this metadata is essential for preservation within the repository. Business rules codify the application of preservation strategies and document repository policies, services, charges, and roles. Retention periods, disposition, risk assessment, permanence ratings, schedules for media refreshment, and so on are pertinent to objects but are not actual properties of Objects. A single exception was made for the level of preservation treatment to be accorded an object (preservationLevel) because this was felt to be critical information for any preservation repository. A more thorough treatment of business rules could be added to the data model by defining a Rules entity similar to Rights, although this is not included in the current revision.

Objects

Object Entity

The Object entity aggregates information about a digital object held by a preservation repository and describes those characteristics relevant to preservation management.

The only mandatory semantic units that apply to all categories of Object (Intellectual Entity, Representation, File, and Bitstream) are *objectIdentifier* and *objectCategory*.

Intellectual Entity
A set of content that is considered a single intellectual unit for purposes of management and description: for example, a particular book, map, photograph, database, or piece of hardware or software. An Intellectual Entity can include other Intellectual Entities; for example, a web site can include a web page; a web page can include an image. An Intellectual Entity may have one or more digital representations.

Representation:
A digital or physical Object instantiating or embodying an Intellectual Entity. A digital representation is the set of stored digital files and structural metadata needed to provide a complete and reasonable rendition of the Intellectual Entity. A physical representation is an item such as a manuscript, video cassette, or printed document.

File:
A named and ordered sequence of bytes that is known to an operating system.

Bitstream:
Contiguous or non-contiguous data within a file that has meaningful properties for preservation purposes.

Entity properties

- Can be associated with one or more Rights statements.
- Can participate in one or more Events.
- Links between entities may be recorded from either direction and need not be bi-directional.

NB: Semantic units are applicable for Intellectual Entities, Representations, Files and Bitstreams unless otherwise indicated.

Entity semantic units

1.1 objectIdentifier (M, R)
  1.1.1 objectIdentifierType (M, NR)
  1.1.2 objectIdentifierValue (M, NR)
1.2 objectCategory (M, NR)
1.3 preservationLevel (O, R) [Intellectual Entity, Representation, File]
  1.3.1 preservationLevelType (O, NR) [Intellectual Entity, Representation, File]
  1.3.2 preservationLevelValue (M, NR) [Intellectual Entity, Representation, File]
  1.3.3 preservationLevelRole (O, NR) [Intellectual Entity, Representation, File]
  1.3.4 preservationLevelRationale (O, R) [Intellectual Entity, Representation, File]
  1.3.5 preservationLevelDateAssigned (O, NR) [Intellectual Entity, Representation, File]
1.4 significantProperties (O, R)
  1.4.1 significantPropertiesType (O, NR)
  1.4.2 significantPropertiesValue (O, NR)
  1.4.3 significantPropertiesExtension (O, R)
1.5 objectCharacteristics (M, R) [File, Bitstream]
  1.5.1 compositionLevel (O, NR) [File, Bitstream]
  1.5.2 fixity (O, R) [File, Bitstream]
    1.5.2.1 messageDigestAlgorithm (M, NR) [File, Bitstream]
    1.5.2.2 messageDigest (M, NR) [File, Bitstream]
    1.5.2.3 messageDigestOriginator (O, NR) [File, Bitstream]
  1.5.3 size (O, NR) [File, Bitstream]
  1.5.4 format (M, R) [File, Bitstream]
    1.5.4.1 formatDesignation (O, NR) [File, Bitstream]
      1.5.4.1.1 formatName (M, NR) [File, Bitstream]
      1.5.4.1.2 formatVersion (O, NR) [File, Bitstream]
    1.5.4.2 formatRegistry (O, NR) [File, Bitstream]
      1.5.4.2.1 formatRegistryName (M, NR) [File, Bitstream]
      1.5.4.2.2 formatRegistryKey (M, NR) [File, Bitstream]
      1.5.4.2.3 formatRegistryRole (O, NR) [File, Bitstream]
    1.5.4.3 formatNote (O, R) [File, Bitstream]
  1.5.5 creatingApplication (O, R) [File, Bitstream]
    1.5.5.1 creatingApplicationName (O, NR) [File, Bitstream]
    1.5.5.2 creatingApplicationVersion (O, NR) [File, Bitstream]
    1.5.5.3 dateCreatedByApplication (O, NR) [File, Bitstream]
    1.5.5.4 creatingApplicationExtension (O, R) [File, Bitstream]
  1.5.6 inhibitors (O, R) [File, Bitstream]
    1.5.6.1 inhibitorType (M, NR) [File, Bitstream]
    1.5.6.2 inhibitorTarget (O, R) [File, Bitstream]
    1.5.6.3 inhibitorKey (O, NR) [File, Bitstream]
  1.5.7 objectCharacteristicsExtension (O, R) [File, Bitstream]
1.6 originalName (O, NR) [Intellectual Entity, Representation, File]
1.7 storage (O, R) [Representation, File, Bitstream]
  1.7.1 contentLocation (O, NR) [Representation, File, Bitstream]
    1.7.1.1 contentLocationType (M, NR) [Representation, File, Bitstream]
    1.7.1.2 contentLocationValue (M, NR) [Representation, File, Bitstream]
  1.7.2 storageMedium (O, NR) [Representation, File, Bitstream]

1.8 signatureInformation (O, R) [File, Bitstream]
  1.8.1 signature (O, R) [File, Bitstream]
    1.8.1.1 signatureEncoding (M, NR) [File, Bitstream]
    1.8.1.2 signer (O, NR) [File, Bitstream]
    1.8.1.3 signatureMethod (M, NR) [File, Bitstream]
    1.8.1.4 signatureValue (M, NR) [File, Bitstream]
    1.8.1.5 signatureValidationRules (M, NR) [File, Bitstream]
    1.8.1.6 signatureProperties (O, R) [File, Bitstream]
    1.8.1.7 keyInformation (O, NR) [File, Bitstream]
  1.8.2 signatureInformationExtension (O, R) [File, Bitstream]
1.9 environmentFunction (O, R) [Intellectual Entity of type environment]
  1.9.1 environmentFunctionType (M, NR) [Intellectual Entity of type environment]
  1.9.2 environmentFunctionLevel (M, NR) [Intellectual Entity of type environment]
1.10 environmentDesignation (O, R) [Intellectual Entity of type environment]
  1.10.1 environmentName (M, NR) [Intellectual Entity of type environment]
  1.10.2 environmentVersion (O, NR) [Intellectual Entity of type environment]
  1.10.3 environmentOrigin (O, NR) [Intellectual Entity of type environment]
  1.10.4 environmentDesignationNote (O, R) [Intellectual Entity of type environment]
  1.10.5 environmentDesignationExtension (O, R) [Intellectual Entity of type environment]
1.11 environmentRegistry (O, R) [Intellectual Entity of type environment]
  1.11.1 environmentRegistryName (M, NR) [Intellectual Entity of type environment]
  1.11.2 environmentRegistryKey (M, NR) [Intellectual Entity of type environment]
  1.11.3 environmentRegistryRole (O, NR) [Intellectual Entity of type environment]
1.12 environmentExtension (O, R) [Intellectual Entity of type environment]
1.13 relationship (O, R)
  1.13.1 relationshipType (M, NR)
  1.13.2 relationshipSubType (M, NR)
  1.13.3 relatedObjectIdentifier (M, R)
    1.13.3.1 relatedObjectIdentifierType (M, NR)
    1.13.3.2 relatedObjectIdentifierValue (M, NR)
    1.13.3.3 relatedObjectSequence (O, NR)
  1.13.4 relatedEventIdentifier (O, R)
    1.13.4.1 relatedEventIdentifierType (M, NR)
    1.13.4.2 relatedEventIdentifierValue (M, NR)
    1.13.4.3 relatedEventSequence (O, NR)
  1.13.5 relatedEnvironmentPurpose (O, R)
  1.13.6 relatedEnvironmentCharacteristic (O, NR)
1.14 linkingEventIdentifier (O, R)
  1.14.1 linkingEventIdentifierType (M, NR)
  1.14.2 linkingEventIdentifierValue (M, NR)
1.15 linkingRightsStatementIdentifier (O, R)
  1.15.1 linkingRightsStatementIdentifierType (M, NR)
  1.15.2 linkingRightsStatementIdentifierValue (M, NR)

1.1 objectIdentifier

Semantic components

1.1.1 objectIdentifierType
1.1.2 objectIdentifierValue

Definition

A designation used to identify the Object uniquely within the preservation repository system in which it is stored

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Mandatory	Mandatory	Mandatory

Rationale

Each object held in the preservation repository must have a unique identifier to allow other entities to refer to it and to relate it to descriptive, technical, and other metadata unambiguously.

Creation/Maintenance notes

An identifier may be created by the repository system at the time of ingest, or it may be created or assigned outside of the repository and submitted with an object as metadata. Similarly, identifiers can be generated automatically or manually.

Usage notes

The **objectIdentifier** is mandatory for all Objects stored. The **objectIdentifier** is repeatable in order to allow both repository-assigned and externally-assigned identifiers to be recorded. See “Creation/Maintenance” note above. Primary

identifiers must be unique within the repository. They may be preexisting, and in use in other digital object management systems. Ideally, secondary identifiers should also be unique but sometimes this is not possible (e.g., if the values are inherited from a legacy system which did not enforce this or only identified items at a higher level). Identifiers for each item must be sufficient to identify the item uniquely at the appropriate level of aggregation. For example, an Intellectual Entity that represents all books in the same edition could use an ISBN but this would be insufficient to identify a particular copy of that book. A preservation repository needs to know both the type of object identifier and the value. If the value itself contains the identifier type (e.g., “oai:lib.uchicago.edu:1”), the identifier type does not need to be recorded explicitly. Similarly, if the repository uses only one type of identifier, the type can be assumed and does not need to be recorded. A persistent identifier should be used, but the particular identifier scheme is an implementation-specific decision.

Table of Contents

1.1.1 objectIdentifierType

Semantic components

None

Definition

A designation of the domain within which the object identifier is unique.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not-repeatable	Not-repeatable	Not-repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	ISBN (Intellectual Entity DOI (Intellectual Entity) DLC DRS hdl:4263537	DLC DRS hdl:4263537	DLC DRS hdl:4263537

Rationale

Identifier values cannot be assumed to be unique across domains; the combination of **objectIdentifierType** and **objectIdentifierValue** should ensure uniqueness.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at http://id.loc.gov/vocabulary/identifiers.html.

Usage notes

The type of the identifier may be implicit within the repository as long as it can be explicitly communicated when the digital object is disseminated outside of it.

Table of Contents

1.1.2 objectIdentifierValue

Semantic components

None

Definition

The value of the objectIdentifier.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	0 00 221804-6 0000000312	IU2440 WAC1943.56 AMNH CD269/CD269/70/10 1001/dig/pres/2004-02 http://nrs.harvard.edu/urn-3:FHCL.Loeb:sa	IU2440-1 IU2440-2

Data Constraint

None

Table of Contents

1.2 objectCategory

Semantic components

None

Definition

The category of object to which the metadata applies.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	intellectual entity representation	file	bitstream

Rationale

Preservation repositories are likely to treat different categories of objects (Intellectual Entities, Representations, Files, and Bitstreams) differently in terms of metadata and data management functions, it is therefore important to differentiate between the categories.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at http://id.loc.gov/vocabulary/preservation/objectCategory.html .

Usage notes

A filestream should be considered a file.

Table of Contents

1.3 preservationLevel

Semantic components

Definition

Information indicating the decision or policy on the set of preservation functions to be applied to an object and the context in which the decision or policy was made.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Not applicable
Repeatability	Repeatable	Repeatable	
Obligation	Optional	Optional	

Rationale

Some preservation repositories will offer multiple preservation options depending on factors such as the value or uniqueness of the material, the "preservability" of the format, the amount the customer is willing to pay, etc. In such circumstances the preservationLevelValue that applies may need to be directly associated with an Object. The choice of a particular preservation option for an object may also require further explanation. This can depend on the preservation functions expected to be applied to the object (which can be described by assigning a **preservationLevelType**) and/or the context in which a set of preservation options is applicable (which can be described by assigning a preservationLevelRole). The distinction between **preservationLevelType** and preservationLevelRole can be illustrated by examples. One possible preservation level type is "Bit preservation level". This might have values of 'Low', 'Medium' or 'High', where, for example, in 2015 technology examples for:

- ‘Low’ means ordinary on-site backup
- ‘Medium’ means two copies on different media types with a minimum of 150 km distance between the stored copies, with separate checksums that are integrity checked annually
- ‘High’ means solutions a minimum of 5 independent copies on a variety of storage media distributed over different organizations in several continents with quarterly integrity checks.

The **preservationLevelRole** can then be used to distinguish, for example, if this level is an aim or has actually been achieved. It is also possible to assign the rationale for choosing the value (which can be described by adding a **preservationLevelRationale**).

Data Constraint

Container

Creation/Maintenance notes

The preservation level may be assigned by the repository or requested by the depositor and submitted as metadata. The repository may also choose to record additional metadata indicating the context for the assignment of the preservation level.

Usage notes

If the repository offers only a single preservation level or the preservation level can be calculated externally (e.g., based on the information in a technical registry or by the type of collection), this value does not need to be explicitly recorded with Objects within the repository. Application of a particular set of **preservationLevel** semantic units may only cover a single Representation of an object: Representations in other technical forms or serving other functions may have a different preservationLevel applied. The container may be repeated if a preservation level value needs to be recorded in additional contexts (see **preservationLevelRole**,or part of a context (see **preservationLevelType**).

Table of Contents

1.3.1 preservationLevelType

Semantic components

None

Definition

A value indicating the type of preservation functions expected to be applied to the Object for this preservation level.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Repeatability	Not repeatable	Not repeatable	
Obligation	Mandatory	Mandatory	
	Bit preservation	Bit preservation	
Example	Logical/functional Preservation	Logical/functional Preservation	

Rationale

Digital preservation functionality can be typed to express various aspects of the preservation. For instance bit-safety can represent a preservation level type, where values of this type can be used to express degree of replication and independence between copies. Likewise logical preservation can represent a type, where values of this type express whether functional preservation is done via format migration, emulation etc.

Data Constraint

Value should be taken from a controlled vocabulary.

Creation/Maintenance notes

The **preservationLevelType** may be assigned by the repository or requested by the depositor and submitted as metadata.

Usage notes

Only one **preservationLevelType** may be recorded per **preservationLevel** container. If a further **preservationLevelType** applies to the Object in a different context, a separate **preservationLevel** container should be repeated.

Table of Contents

1.3.2 preservationLevelValue

Semantic components

None

Definition

A value indicating the set of preservation functions expected to be applied to the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Repeatability	Not repeatable	Not repeatable	
Obligation	Mandatory	Mandatory	
Examples	For preservationLevelType “Logical Preservation”: Migration Emulation	For preservationLevelType “Bit Preservation”: Low (e.g. backup) Medium (e.g. min. 2 copies and integrity check) High (e.g. min 5 copies, integrity check, and high independence	

Rationale

Allows a value to be assigned to the **preservationLevel**.

Data Constraint

Value should be taken from a controlled vocabulary. If **preservationLevelType** and/or **preservationLevelRole** are used, then the available controlled vocabulary should be dependent on the values set for each of these types.

Creation/Maintenance notes

The preservation level may be assigned by the repository or requested by the depositor and submitted as metadata.

Usage notes

Only one **preservationLevelValue** may be recorded per **preservationLevel** container. If a further preservationLevelValue applies to the Object in a different context, a separate **preservationLevel** container should be repeated.

Table of Contents

1.3.3 preservationLevelRole

Semantic components

None

Definition

A value indicating the context in which a set of preservation options is applicable.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Repeatability	Not repeatable	Not repeatable	
Obligation	Optional	Optional	

Rationale

Repositories may assign **preservationLevelValues** in different contexts which must be differentiated, and may need to record more than one context. This allows a distinction, for example, between the intended preservation level and the current achievable preservation level. Note: This is distinct from the **preservationLevelType** which distinguishes the purpose of the **preservationLevel**.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary list is available at: http://id.loc.gov/vocabulary/preservation/preservationLevelRole.htm

Usage notes

This optional semantic unit qualifies the sense or context in which the **preservationLevelValue** in the current preservationLevel container is

applied. For example, a repository may have a legislated obligation to “fully preserve” object X (which is of format F) but is presently only capable of preserving objects of format F at a “bit-level”. The repository may need to record both the required or intended level of preservation (e.g. **preservationLevelRole** = “requirement”) and the current capability (e.g. **preservationLevelRole** = “capability”). In transferring custody of material from one repository to another, it may also be important for the receiving repository to know the sense in which **preservationLevelValue** should be understood. A receiving repository may not need to know a “capability” preservation level of which the transferring repository was capable (as this will have little bearing on its own capabilities), but it needs to know any preservation level “requirements” for material for which it is now taking responsibility. It is good practice to specify **preservationLevelRole** for clarity even if the repository only assigns **preservationLevelValue** in one sense or context. If more than one **preservationLevel** is recorded with the same **preservationLevelType**, **preservationLevelRole** should always be supplied. If more than one sense or context needs to be expressed for the same object, (e.g. both the “requirement” and “capability” are recorded), separate **preservationLevel** containers should be used.

Table of Contents

1.3.4 preservationLevelRationale

Semantic components

None

Definition

The reason a particular **preservationLevelValue** was applied to the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Repeatability	Repeatable	Repeatable	
Obligation	Optional	Optional	
Examples	user pays legislation	defective file bit-level preservation only available for this file	

Rationale

Application of a particular **preservationLevelValue** may require justification, especially if it differs from that usually applied according to repository policy.

Data Constraint

None

Usage notes

This optional semantic unit records the reason for applying the **preservationLevelValue**. This information can be particularly important when the assigned **preservationLevelValue** differs from usual repository policy. For example, a repository may normally assign a **preservationLevelValue** of “full preservation” for JPEG2000 files, but detects that a particular file is defective. This may mean that the repository’s preservation strategy for

JPEG2000 may not be effective for this particular file, so the repository may assign a **preservationLevelValue** of “bit-level preservation” to this file , recording “defective file” as the rationale. Similarly, legislative requirements or contractual agreements may require a higher level of preservation to be assigned to a particular object than would be assigned to that class of object according to usual policy. In this case, the rationale for the assignment may be recorded as “legislation” or “user pays”, for example. **preservationLevelRationale** may be repeated if more than one reason needs to be recorded.

Table of Contents

1.3.5 preservationLevelDateAssigned

Semantic components

None

Definition

The date, or date and time, when a particular **preservationLevelValue** was assigned to the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Repeatability	Not repeatable	Not repeatable	
Obligation	Optional	Optional	
Examples	2007-11-05 2007-11-05T08:15:30-05:00 20080315	2007-11-05 2007-11-05T08:15:30-05:00 20080315	

Rationale

The **preservationLevel** applicable to an object is expected to be reviewed and changed over time, in response to changes in repository preservation requirements, policies, or capabilities relevant to the object. The date that the current **preservationLevelValue** was assigned aids review of decisions.

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Table of Contents

1.4 significantProperties

Semantic components

Definition

Characteristics of a particular object subjectively determined to be important to maintain through preservation actions.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional

Rationale

Objects that have the same technical properties may still differ as to the properties that should be preserved for future presentation or use.

Data Constraint

Container

Creation/Maintenance notes

Significant properties may pertain to all objects of a certain class; for example, the repository can decide that for all PDF files, only the content need be preserved. In other cases for example, for media art, the significant properties may be unique to each individual object. Where values are unique, they must be supplied by the submitter or provided by the curatorial staff of the repsoitory.

Usage notes

At least one of the significantPropertiesValue and **significantPropertiesExtension** subunits must be present if this container is included, or both may be used.

The choice of whether a property is significant is subjective. Some of these properties can be directly measured while others can only be determined subjectively.

For example, a PDF may contain links that are not considered important and JavaScript that is considered important. Or future migrations of a TIFF image may require optimization for line clarity or for color; the option chosen would depend upon a curatorial judgment of the significant properties of the image.

Listing significant properties implies that the repository plans to preserve these properties across time and requires them to survive preservation action acceptably; for example, to be maintained during emulation orafter format migration. It also implies that the repository would note when preservation action results in modification of significant properties.

In practice, significant properties might be used as measures of DATA DICTIONARY preservation success, as part of quality checking the results of a preservation action or evaluating the efficacy of a preservation method. For example, if the listed significant properties are not maintained after application of a particular preservation method, it may indicate a failure of the process or that the method is not well suited to the type of material.

More experience with digital preservation is needed to determine the best ways of representing significant properties in general, and of representing modification of significant properties.

The semantic units included in the **significantProperties** container aim to provide a flexible structure for describing significant properties, allowing general types of aspects, facets or attributes of an object to be declared and to be paired with specific significant details about the object pertaining to that aspect, facet or attribute.

For example, some repositories may define significant properties for objects related to facets of content, appearance, structure, behavior, and context. Examples of facet:detail pairs in this case are as follows (note that each facet:detail pair should be contained in a separate, repeated **significantPropertiescontainer**): **significantPropertiesType** = “content”

significantPropertiesValue = “all textual content and images”

significantPropertiesType = “behavior” significantPropertiesValue = “editable”

Other repositories may choose to describe significant properties at a more granular attribute level; for example:

significantPropertiesType = “page count”

significantPropertiesValue = “7”

significantPropertiesType = “page width”

significantPropertiesValue = “210 mm”

Further work on determining and describing significant properties may yield more detailed schemes to facilitate general description.

Representing modification of significant properties as a result of preservation action also requires further work. One possible way involves the use of Object and Event information: Object A has the significant properties volume and timing, which are recorded as **significantProperties** of A. In migrated version B, the timing is modified, which is noted in the **eventOutcome** of the migration Event. Only volume is listed as a significant property of B.

Table of Contents

1.4.1 significantPropertiesType

Semantic components

None

Definition

The aspect, facet, or attribute of an object which is being recorded as a significant property.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Examples	content structure behavior page count page width typeface hyperlinks image count	content structure behavior page count page width typeface	[for an embedded image] color space

Rationale

Repositories may choose to describe significant properties based on a particular aspect or attribute of an object.

Data Constraint

Container

Usage notes

This semantic unit is optional and may be used as part of a facet:detail pair with **significantPropertiesValue.**

Table of Contents

1.4.2 significantPropertiesValue

Semantic components

None

Semantic components

None

Definition

Description of a characteristic of a particular object which has been determined subjectively to be important to maintain through preservation actions.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	At least one of the **significantProperties-Value** and **significantProperties-Extension** subunits must be present if this container is included, or both may be used.	At least one of the **significantProperties-Value** and **significantProperties-Extension** subunits must be present if this container is included, or both may be used.	At least one of the **significantProperties-Value** and **significantProperties-Extension** subunits must be present if this container is included, or both may be used.
Examples	[for a Web page containing animation that is not considered essential] Content only [For detail associated with a significantPropertiesType of “behavior”] hyperlinks traversable	[for a word processed document with embedded links that are not considered essential] Content only. [For detail associated with a significantPropertiesType of “behavior”] editable [For detail associated with a significantPropertiesType of	[for a PDF with an embedded graph, where the lines’ color determines the lines’ meaning] Color. [For detail associated with a significantPropertiesType of “appearance”] Color.

“page width”] 210
mm

Rationale

Repositories may choose to describe significant properties based on a particular aspect or attribute of an object.

Data Constraint

None

Usage notes

If facet:detail pairs are used, the content of **significantPropertiesValue** should describe the significant properties of the object relevant to the aspect, facet, or attribute declared in the **significantPropertiesType** with which it is paired. If facet:detail pairs are not used, **significantPropertiesValue** may be used to freely to describe any characteristic of an object. **significantPropertiesValue** is not repeatable. Multiple significant properties should be described in separate, repeated **significantProperties** container units.

Table of Contents

1.4.3 significantPropertiesExtension

Semantic components

Defined externally

Definition

A container to include semantic units defined outside of PREMIS for significant properties.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Not repeatable	Repeatable
Obligation	At least one of the **significantProperties-Value** and **significantProperties-Extension** subunits must be present if the **significantProperties** container is included, or both may be used	At least one of the **significantProperties-Value** and **significantProperties-Extension** subunits must be present if the **significantProperties** container is included, or both may be used	At least one of the **significantProperties-Value** and **significantProperties-Extension** subunits must be present if the **significantProperties** container is included, or both may be used

Rationale

There may be a need to replace or extend PREMIS defined units.

Usage notes

For more granularity or use of externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata scheme may be included in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema must be provided. See further guidance in “Extensibility”.

All of this semantic unit's subunits are optional. At least one of the **significantPropertiesValue** and **significantPropertiesExtension** subunits must be present if the **significantProperties** container is included.

If the **significantPropertiesExtension** container needs to be associated explicitly with any PREMIS subunit under significantProperties, the container **significantProperties** is repeated. Also, if extensions from different external schemas are needed, **significantProperties** should be repeated.

It is recommended to give information about the metadata used in **significantPropertiesExtension** including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

1.5 objectCharacteristics

Semantic components

Definition

Information used to verify whether an object has been altered in an undocumented or unauthorized way.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable (see usage note)	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Rationale

There are some important technical properties that apply to objects of any format. Detailed definition of format-specific properties is outside the scope of this Data Dictionary, although such properties may be included within objectCharacteristicsExtension.

Data Constraint

Container

Creation/Maintenance notes

Significant properties may pertain to all objects of a certain class; for example, the repository can decide that for all PDF files, only the content need be preserved. In other cases for example, for media art, the significant properties may be unique to each individual object. Where values are unique, they must be supplied by the submitter or provided by the curatorial staff of the repsoitory.

Usage notes

To perform a fixity check, a message digest calculated at some earlier time is compared with a message digest calculated at a later time. If the digests are the same, the object was not altered in the interim. (Note that the terms “message digest” and “checksum” are commonly used interchangeably. However, the term “checksum” is more correctly used for the product of a cyclical redundancy check (CRC), whereas the term “message digest” refers to the result of a cryptographic hash function, which is what is referred to here.)The act of performing a fixity check and the date it occurred would be recorded as an Event. The result of the check would be recorded as the eventOutcome. Therefore, only the **messageDigestAlgorithm** and **messageDigest** need to be recorded as objectCharacteristics for future comparison.

Representation level: it could be argued that if a representation consists of a single file or if all the files comprised by a representation are combined (e.g., zipped) into a single file, then a fixity check could be performed on the representation. However, in both cases the fixity check is actually being performed on a file, which in this case happens to be coincidental with a representation. Bitstream level: message digests can be computed for bitstreams although they are not as common as with files. For example, the JPX format, which is a JPEG2000 format, supports the inclusion of MD5 or SHA-1 message digests in internal metadata that was calculated on any range of bytes of the file. For more information, see the special topic on “Fixity, integrity, authenticity,”.

Table of Contents

1.5.1 compositionLevel

Semantic components

None

Definition

An indication of whether the Object is subject to one or more processes of decoding or unbundling.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		0 1 2 Unknown	0 1 2 Unknown

Rationale

A file or bitstream can be encoded with compression, encryption, etc., or bundled with other files or bitstreams into larger packages. Knowing the order in which these actions are taken is important if the original Object or Objects must be recovered.

Data Constraint

Non-negative integers (or “unknown”)

Creation/Maintenance notes

Composition level will generally be supplied by the repository, which should attempt to supply this value automatically. If the object was created by the

repository, the creating routine knows the composition level and can supply this metadata. If the object is being ingested by the repository, repository programs will have to attempt to identify the composition level from the object itself or from externally supplied metadata.

Usage notes

A file or bitstream can be subject to multiple encodings that must be decoded in reverse order (highest to lowest). For example, file A may be compressed to create file B, which is encrypted to create file C. To recreate a copy of the base file A, one would have to decrypt file C to create file B and then decompress file B to create file A. A **compositionLevel** of zero indicates that the object is a base object and not subject to further decoding, while a level of 1 or higher indicates that one or more decodings must be applied. Numbering goes lowest to highest (first encoded = 0). 0 is base object; 1-n are subsequent encodings. The **compositionLevel** should be set whenever possible, however it is permitted to omit (or use "unknown") if it not possible to calculate this. Use 0 if there is only one **compositionLevel**. When multiple file objects are bundled together as filestreams within a package file object (e.g., a ZIP file), the individual filestream objects are not composition levels of the package file object. They should be considered separate objects, each with their own composition levels. For example, two encrypted files zipped together and stored in an archive as one file object would be described as three separate objects, each with its own associated metadata. The storage location of the two inner objects would point to the ZIP file, but the ZIP file itself would have only a single composition level (of zero) whose format would be "zip." See "Object characteristics and composition level".

Table of Contents

1.5.2 fixity

Semantic components

Definition

Information used to verify whether an object has been altered in an undocumented or unauthorized way.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable (see usage note)	Applicable	Applicable (see usage note)
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Data Constraint

Container

Creation/Maintenance notes

Automatically calculated and recorded by repository.

Usage notes

To perform a fixity check, a message digest calculated at some earlier time is compared with a message digest calculated at a later time. If the digests are the same, the object was not altered in the interim. (Note that the terms “message digest” and “checksum” are commonly used interchangeably. However, the term “checksum” is more correctly used for the product of a cyclical redundancy check (CRC), whereas the term “message digest” refers to the result

of a cryptographic hash function, which is what is referred to here.)The act of performing a fixity check and the date it occurred would be recorded as an Event. The result of the check would be recorded as the **eventOutcome**. Therefore, only the **messageDigestAlgorithm** and **messageDigest** need to be recorded as objectCharacteristics for future comparison. Representation level: it could be argued that if a representation consists of a single file or if all the files comprised by a representation are combined (e.g., zipped) into a single file, then a fixity check could be performed on the representation. However, in both cases the fixity check is actually being performed on a file, which in this case happens to be coincidental with a representation. Bitstream level: message digests can be computed for bitstreams although they are not as common as with files. For example, the JPX format, which is a JPEG2000 format, supports the inclusion of MD5 or SHA-1 message digests in internal metadata that was calculated on any range of bytes of the file. For more information, see the special topic on “Fixity, integrity, authenticity,”.

Table of Contents

1.5.2.1 messageDigestAlgorithm

Semantic components

None

Definition

The specific algorithm used to construct the message digest for the digital object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		MD5 Adler-32 HAVAL SHA-1 SHA-256 SHA-384 SHA-512 TIGER WHIRLPOOL	MD5 Adler-32 HAVAL SHA-1 SHA-256 SHA-384 SHA-512 TIGER WHIRLPOOL

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary list is available at: http://id.loc.gov/vocabulary/preservation/cryptographicHashFunctions.html.

Table of Contents

1.5.2.2 messageDigest

Semantic components

None

Definition

The output of the message digest algorithm.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		7c9b35da4f2ebd436f1cf887e5a39b3a257edf4a22be3c955ac49da2e2107b67a192441956	7c9b35da4f2ebd436f1cf88e5a39b3a257edf4a22be3c955ac49da2e2107b67a192441956

Rationale

This must be stored so that it can be compared in future fixity checks.

Data Constraint

None

Table of Contents

1.5.2.3 messageDigestOriginator

Semantic components

None

Definition

The Agent that created the original message digest that is compared in a fixity check.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		DRS A0000978	DRS A0000978

Rationale

A preservation repository may ingest files that have had message digests calculated by the submitter; checking these ensures that the file as received is the same as the file as sent. The repository may also ingest files that do not have message digests, and so must calculate the initial value upon ingest. It can be useful to know what Agentcalculated the initial value of the message digest.

Data Constraint

None

Creation/Maintenance notes

If the calculation of the initial message digest is treated by the repository as an Event, this information could be obtained from an Event record.

Usage notes

The originator of the message digest could be represented by a string representing the Agent (e.g., “DRS” referring to the archive itself) or a pointer to an Agent description (e.g., “A0000978” taken here to be an **agentIdentifierValue**)

Table of Contents

1.5.3 size

Semantic components

None

Definition

The size in bytes of the file or bitstream stored in the repository.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		2038937	2038937

Rationale

Size is useful for ensuring the correct number of bytes from storage has been retrieved and that an application has enough room to move or process files. It might also be used when billing for storage.

Data Constraint

Integer

Creation/Maintenance notes

Automatically obtained by the repository.

Usage notes

Defining this semantic unit as size in bytes makes it unnecessary to record a unit of measurement. However, for the purpose of data exchange the unit of measurement should be stated or understood by both partners.

Table of Contents

1.5.4 format

Semantic components

Definition

Identification of the format of a file or bitstream where format is defined as the organization of digital information according to preset specifications.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Mandatory	Mandatory

Rationale

Many preservation activities depend on detailed knowledge about the format of the digital object. An accurate identification of format is essential. The identification provided, whether by name or pointer to a format registry, should be sufficient to associate the object with more detailed format information.

Data Constraint

Container

Creation/Maintenance notes

The format of a file or bitstream should be ascertained by the repository on ingest. Even if this information is provided by the submitter, directly in metadata or indirectly via the filename extension, recommended practice is

to identify the format independently by parsing the file when possible. If the format cannot be identified at the time of ingest, it is valid to record that it is unknown, but the repository should subsequently make an effort to identify the format, even if manual intervention is required.

Usage notes

A bitstream embedded within a file may have different characteristics from the larger file. For example, a bitstream in LaTeX format could be embedded within an SGML file, or multiple images using different colorspaces could be embedded within a TIFF file. **format** must be recorded for every object. When the bitstream format can be recognized by the repository and the repository might want to treat the bitstream differently from the embedding file for preservation purposes, **format** can be recorded for embedded bitstreams. At least one subunit (i.e. either **formatDesignation** or **formatRegistry**) must be present if this container is included, or both may be used. If the subunit (**formatDesignation** or **formatRegistry**) needs to be repeated, the entire format container is repeated. This allows for association of format designation with a particular set of format registry information. For example, if the precise format cannot be determined and two **format** designations are recorded, each is given within a separate **format** container. In such cases the **formatNote** element can be used to distinguish between the cases where either (i) it is known that the file complies with multiple format definitions or (ii) it is known that the file complies with one of these formats but there is insufficient knowledge to distinguish between them. The **format** container may also be repeated for multiple format registry entries. If a file or bitstream is in an unknown format then a **formatDesignation** element should be added with formatName set to unknown. See “Format information”.

Table of Contents

1.5.4.1 formatDesignation

Semantic components

1.5.4.1.1 formatName
1.5.4.1.2 formatVersion

Definition

An identification of the format of the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		At least one subunit must be present if this container is included.	At least one subunit must be present if this container is included.

Data Constraint

Container

Usage notes

Either **formatDesignation** or at least one instance of **formatRegistry** is required. Both may be included. The most specific format (or format profile) should be recorded. A repository (or format registry) may wish to use multipart format names (e.g., “TIFF_GeoTIFF” or “WAVE_MPEG_BWF”) to achieve this specificity. For any given file or bitstream, the most specific format identified by the repository should be recorded. A restricted or modified version of a format is considered more specific than the format; for example, GeoTIFF is more specific than TIFF; BWF is more specific than WAVE. If a file or bitstream conforms to more than one format of equal specificity, each should be recorded in separate **format** containers.

Table of Contents

1.5.4.1.1 formatName

Semantic components

None

Definition

A commonly accepted name for the format of the file or bitstream.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		Text/sgml image/tiff/geotiff AdobePDF DES PGP base64 unknown	LaTex

Data Constraint

Value should be taken from a controlled vocabulary.

Usage notes

For unidentified formats, **formatName** may be recorded as “unknown”.
Whenever possible, controlled vocabularies for **formatName** should come from format or technical registries.

Table of Contents

1.5.4.1.2 formatVersion

Semantic components

None

Definition

The version of the format named in formatName.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		6.0 2003	7.1

Rationale

Many authority lists of format names are not granular enough to indicate version, for example, MIME Media types.

Data Constraint

None

Usage notes

If the format is versioned, **formatVersion** should be recorded where applicable. It can be either a numeric or chronological designation.

Table of Contents

1.5.4.2 formatRegistry

Semantic components

1.5.4.2.1 formatRegistryName
1.5.4.2.2 formatRegistryKey
1.5.4.2.3 formatRegistryRole

Definition

Identifies and/or gives further information about the format by reference to an entry in a format registry.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		At least one subunit must be present if this container is included.	At least one subunit must be present if this container is included.

Data Constraint

Container

Usage notes

Either **formatDesignation** or at least one instance of **formatRegistry** is required. If more than one **formatRegistry** needs to be recorded the format container should be repeated to include each additional set of **formatRegistryinformation**.

Table of Contents

1.5.4.2.1 formatRegistryName

Semantic components

None

Definition

A designation identifying the referenced format registry.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		PRONOM http://www.nationalarchives.gov.uk/pronom Representation Information Registry Repositor	PRONOM http://www.nationalarchives.gov.uk/pronom Representation Information Registry Repositor

Data Constraint

None

Usage notes

This can be a formal name, internally used name, or URI.

Table of Contents

1.5.4.2.2 formatRegistryKey

Semantic components

None

Definition

The unique key used to reference an entry for this format in the specified format registry.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		fmt/155	fmt/155

Data Constraint

None

Table of Contents

1.5.4.2.3 formatRegistryRole

Semantic components

None

Definition

The purpose or expected use of the registry.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		Specification Validation profile	Specification Validation profile

Rationale

The same **format** may be defined in different registries for different purposes. For example, one registry may give detailed format specifications while another has profile information / information about software support and dependencies. If multiple registries are recorded, this semantic unit can be used to distinguish among them.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/formatRegistryRole.html.

Table of Contents

1.5.4.3 formatNote

Semantic components

None

Definition

Additional information about format.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Examples		tentative identification disjunction multiple format identification found	tentative identification disjunction multiple format identification found

Rationale

Qualifying information may be needed to supplement format designation and registry information or to record a status for identification.

Data Constraint

None

Usage notes

The **formatNote** may contain free text, a reference pointer, or a value from a controlled list.

Table of Contents

1.5.5 creatingApplication

Semantic components

Definition

Information about the application that created the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Rationale

Information about the creating application, including the version of the application and the date the file was created, can be useful for problem solving purposes. For example, it is not uncommon for certain versions of software to be known for causing conversion errors or introducing artifacts. It is also useful to determine whatrendering software is available for the digital object. For example, if you know that the Distiller program created the PDF file, you know it will be renderable with (among other programs) Adobe Reader.

Data Constraint

Container

Creation/Maintenance notes

If the object was created by the repository, assignment of creating application information should be straightforward.If the object was created outside the repository, it is possible this information could be supplied by the depositor. It might also be extracted from the file itself; the name of the creating application is often embedded within the file.

Usage notes

This semantic unit applies both to objects created outside of the repository and subsequently ingested, and to objects created by the repository, for example, through migration Events. The **creatingApplication** container is repeatable if more than one application processed the object in turn. For example, a file could be created by Microsoft Word and later turned into a PDF using Adobe Acrobat. Details of both the Word and Acrobat applications may be recorded. However, if both files are stored in the repository, each fileshould be completely described as an Object entity and linked by using relationship information with a **relationshipType** “derivation." The container may also be repeated to record the creating application before the object was ingested as well as the creating application used as part of the ingest process. For example, an HTML filewas created pre-ingest using Dreamweaver, and the Web crawler Heritrix then captured a snapshot of the files as part of ingest. **creatingApplication** is a frequently used semantic unit; however, with the introduction of more detailed environment capability, best practice should be to use an environment Object rather than this semantic unit. The use of environment Objects allows repositories to build complex relationships between objects and their creating or rendering environments. It also allows for the recording of relationships between environments themselves, for example, to expose dependencies between multiple components of complex software stacks. For more information on using environment Objects, see section “Special Topics”. However, unlike environment Objects, **creatingApplication** enables the description of different creating applications at different composition levels of the Object. It is also supported for backward compatibility. Rather than having each repository record this locally, it is preferable to share this information using environment registries.

Table of Contents

1.5.5.1 creatingApplicationName

Semantic components

None

Definition

A designation for the name of the software program that created the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		MS Word	MS Excel

Data Constraint

None

Usage notes

The **creatingApplication** is the application that created the object in its current format, not the application that created the copy written to storage. For example, if a document is created by Microsoft Word and subsequently copied to archive storage by a repository's Ingest program, the **creatingApplication** is MS Word, not the Ingest program.

Table of Contents

1.5.5.2 creatingApplicationVersion

Semantic components

None

Definition

The version of the software program that created the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		2000	1.4

Data Constraint

None

Table of Contents

1.5.5.3 dateCreatedByApplication

Semantic components

None

Definition

The actual or approximate date and time the object was created.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		2000-12-01 20030223T151047	2000-12-01 20030223T151047

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Usage notes

Use the most precise date available. This is the date the object was created by the creating application, not the date any copy was made externally or by the repository. For example, if a file is created by Microsoft Word in 2001 and two copies are made in 2003, the **dateCreatedByApplication** of all three files is 2001. The date a file is written to storage can be recorded as an Event.If the object itself contains internal creation and modification dates, the modification date should be used as **dateCreatedByApplication**. The creation date, in PREMIS terms, of an Object is the time that it was last modified; that is, the last time the document was saved. This is discussed in the 1:1 Principle INSERTLINKHERE section. If the application is a Web harvester capturing an object at a point intime, use the date captured as the creation date.

Table of Contents

1.5.5.4 creatingApplicationExtension

Semantic components

Defined externally

Definition

Creating application information using semantic units defined externally to PREMIS.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Rationale

There may be a need to supplement or replace PREMIS defined units.

Data Constraint

Container

Usage notes

For more granularity or to use externally defined semantic units, extensibility is provided. This could include a reference to an external technical registry describing the creating application. Either local semantic units or metadata using another specified metadata scheme may be included instead of or in addition to PREMIS-defined semantic units. When using an externally defined schema, a reference to that schema must be provided. See further guidance on “Extensibility”. If multiple extensions are needed (e.g., for different purposes), the container **creatingApplication** is repeated. Also, if extensions from different external schemas are needed, **creatingApplication** should be

repeated. It is recommended to give information about the metadata used in **creatingApplicationExtension** including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

1.5.6 inhibitors

Semantic components

Definition

Features of the object that inhibit access, use, or migration.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Rationale

Format information may indicate whether a file is encrypted, but the nature of the encryption also must be recorded, as well as the access key.

Data Constraint

Container

Creation/Maintenance notes

Inhibitors are more likely to be present on an object ingested by the repository than applied by the repository itself. It is hard to detect during ingest if a file is encrypted, but this may be known from the context in which it is ingested. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.

Usage notes

Some file formats allow encryption for embedded bitstreams. Some file formats such as PDF use passwords to control access to content or specific functions. Although this is actually implemented at the bitstream level, for preservation purposes it is effectively managed at the file level; that is, passwords would not be recorded for individually addressable bitstreams. For certain types of inhibitor keys, more granularity may be requiredin local applications. If the inhibitor key information is identical to key information in digital signatures, use those semantic units.

Table of Contents

1.5.6.1 inhibitorType

Semantic components

None

Definition

The inhibitor method employed.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		DES PGP Blowfish Password protection	DES PGP Blowfish Password protection

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/inhibitorType.html.

Usage notes

Common inhibitors are encryption and password protection. When encryption is used the type of encryption should be specifically indicated, that is, record “DES”, not “encryption”.

Table of Contents

1.5.6.2 inhibitorTarget

Semantic components

None

Definition

Features of the object that inhibit access, use, or migration.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
		All content	All content
Examples		Function: Play	Function: Play
		Function: Print	Function: Print

Data Constraint

Values should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/inhibitorTarget.html.

Usage notes

If not supplied, assume that the target is the content of the object.

Table of Contents

1.5.6.3 inhibitorKey

Semantic components

None

Definition

The decryption key or password

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Examples		[DES decryption key]	[DES decryption key]

Data Constraint

None

Usage notes

The key should be provided if known. However, it is not advisable to actually store the **inhibitorKey** in plain text in an unsecure database.

Table of Contents

1.5.7 objectCharacteristicsExtension

Semantic components

Defined externally

Definition

A container to include semantic units defined outside of PREMIS.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Rationale

There may be a need to replace or extend PREMIS defined units.

Data Constraint

Container

Usage notes

For more granularity or use of externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata scheme may be included in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema must be provided. See further guidance in “Extensibility”, **objectCharacteristicsExtension** is used for additional object characteristics not covered by PREMIS, for instance format-specific metadata that is defined externally. It is not a replacement for units specified in PREMIS.If multiple extensions are needed (e.g., for different purposes), the container **objectCharacteristics** needs to be repeated. Also, if extensions from

different external schemas are needed, objectCharacteristics should be repeated.It is recommended to give information about the metadata used in **objectCharacteristicsExtension** including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

1.6 originalName

Semantic components

None

Definition

The name of the object as submitted to or harvested by the repository, before any renaming by the repository.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Repeatability	Not repeatable	Not repeatable	
Obligation	Optional	Optional	
Examples	"Animal Antics"	N419.pdf	

Rationale

This unit allows preservation system to rename files (for internal management and storage purposes) but still retain the name of the object that was submitted. The name used within the preservation repository may not be known outside of the repository. A depositor might need to request a file by its original name. Also, the repository may need to reconstruct internal links for dissemination.

Data Constraint

None

Creation/Maintenance notes

This value would always be supplied to the repository by the submitter or harvesting application. How much of the file path to preserve would be up to the repository.

Usage notes

This is designed to be used to hold the name supplied by the immediate source before ingest by the repository. This may or may not be the initial name that the object had (e.g., the source could have renamed a file when it ingested it from a previous source and not retained its initial name). In addition it may or may not be the name that the object was called in the immediate source before the repository (e.g., the source system might hold it with one name but rename it upon export to, say, its truly original name). The original name of the object must be designated in the Submission Information Package (SIP) (since the repository cannot in general work it out by other means) although it may or may not be the primary identifier of the object in the SIP. The object may have other names in different contexts. When two repositories are exchanging content, it would be important for the receiving repository to know and record the name of the Intellectual Entity or Representation at the originating repository. In the case of IEs or representations, this may be a directory name.

Table of Contents

1.7 storage

Semantic components

Definition

Information about how and where an entity can be found. For bitstreams this means the location within a file. For files this means the physical location in one or more storage systems. Starting with PREMIS version 3.0, **storage** is applicable to representations. For physical representations such as system disks or printed versions of digital files, this means the location of the physical object, such as a shelf location. For digital representations this means the location of the digital object in a storage system, if all files in the representation can be found in this same location.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable	Repeatable
Obligation		Optional	Optional	Optional

Rationale

It is necessary for a repository to know where to locate objects and to associate the **contentLocation** of digital objects with the **storageMedium**.

Data Constraint

Container

Usage notes

For digital representations and files, the **storage** container should be repeated if there are two or more copies that are identical bit-wise and managed as a

unit, except for the medium on which they are stored. To use this repetition, the copies must have a single **objectIdentifier** and be managed as a single object by the repository. Note that many storage systems (e.g., hierarchical storage management systems, cloud storage providers etc.) will provide only a single reference to an object even though it may store multiple copies. For bitstreams, the **storage** container could be repeated if there is more than one way of getting to the start of the bitstream within a file (e.g., byte offset from start or byte offset from end). For representations of physical Objects, the storage for both physical copies of content (for example, a paper copy of a digital object) and for physical environment instances (for example, a custom computer chip) can be specified. See the “Environment” section in “Special Topics” for more information on physical Environments.

Table of Contents

1.7.1 contentLocation

Semantic components

1.7.1.1 contentLocationType
1.7.1.2 contentLocationValue

Definition

Information needed to retrieve a physical item from its physical storage location or a file from the storage system, or to access a bitstream within a file.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable	Not repeatable
Obligation		Optional	Optional	Optional

Data Constraint

Container

Creation/Maintenance notes

A preservation repository will normally refer to content that it controls. Therefore, it is assumed that the repository will normally assign the **contentLocation**, probably by a program.

Usage notes

If the preservation repository uses the **objectIdentifier** as a handle for retrieving data, **contentLocation** is implicit and does not need to be recorded.

Table of Contents

1.7.1.1 contentLocationType

Semantic components

None

Definition

The means of referencing the location of the content.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory	Mandatory
Examples		Physical storage location Shelfmark RFID number	URI handle NTFS EXT3	Byte offset

Rationale

To understand the meaning of the value it is necessary to know what location scheme is used.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/contentLocationType.html.

Table of Contents

1.7.1.2 contentLocationValue

Semantic components

None

Definition

The reference to the location of the content used by the storage system.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory	Mandatory
Examples		Shel4, box 3 DA592.B76 [Shelfmark] 3054257BF4C21B4000001ABF [RFID number http://wwasearch.loc.gov/107th/200212107035 http://house.gov/langevin hdl:loc.pnp/cph.3b34188 c:\apache2\htdocs\index.htm /home/web/public_html/index.htm	http://wwasearch.loc.gov/107th/200212107035 http://house.gov/langevin hdl:loc.pnp/cph.3b34188 c:\apache2\htdocs\index.htm /home/web/public_html/index.htm	64 [offset from start of file c:\apache2\htdocs\image\logo.gif]

Data Constraint

None

Creation/Maintenance notes

Inhibitors are more likely to be present on an object ingested by the repository than applied by the repository itself. It is hard to detect during ingest if a file is encrypted, but this may be known from the context in which it is ingested. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.

Usage notes

This could be a fully qualified path and filename, or the information used by a resolution system (e.g., a handle), or the native information used by a storage management system. For a bitstream or filestream, this would probably be the reference point and offset of the starting position of the bitstream within a file. It is up to the repository to determine the level of granularity that should be recorded.

Table of Contents

1.7.2 storageMedium

Semantic components

None

Definition

The physical medium on which the object is stored (e.g., magnetic tape, hard disk, CD-ROM, DVD).

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable	Not repeatable
Obligation		Optional	Optional	Optional
Examples		Magnetic tape Hard disk TSM	Magnetic tape Hard disk TSM	Magnetic tape Hard disk TSM

Rationale

The repository needs to know the medium on which an object is stored if it is to know how and when to do media refreshment and media migration.

Data Constraint

Value should be taken from a controlled vocabulary. Example controlled vocabularies are available at: http://id.loc.gov/vocabulary/preservation/storageMedium.html and http://metadataregistry.org/concept/list/vocabulary_id/145.html.

Usage notes

In some cases this can be masked from direct repository management by storage management systems. In such cases responsibility to manage for technological obsolescence of storage media must be delegated to the storage

system. Digital preservation practitioners must maintain responsibility to manage storage media and system obsolescence within the combination of repository and storage systems. Usually this is done at system level well above the details of individual objects. *Storage media are only recorded for digital items. The storage locations of physical items will be managed through traditional estate management, and media management is not part of the digital repository.

Table of Contents

1.8 signatureInformation

Semantic components

1.8.1 signature
1.8.2 signatureInformationExtension

Definition

A container for PREMIS-defined and externally-defined digital signature information, used to authenticate the signer of an object and/or the information contained in the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Rationale

A repository may have a policy of generating digital signatures for files on ingest, or may have a need to store and later validate incoming digital signatures.

Data Constraint

Container

Usage notes

Either **signature** or **signatureInformationExtension** may be used. Use of **signatureInformationExtension** with the schema defined in W3C's XML-Signature Syntax and Processing(http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/) is encouraged when applicable. See the discussion of digital

signatures for more information on use of both PREMIS-defined and externally-defined semantic units.

Table of Contents

1.8.1 signature

Semantic components

Definition

Information needed to use a digital signature to authenticate the signer of an object and/or the information contained in the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Rationale

A repository may have a policy of generating digital signatures for files on ingest, or may have a need to store and later validate incoming digital signatures.

Data Constraint

Container

Creation/Maintenance notes

Inhibitors are more likely to be present on an object ingested by the repository than applied by the repository itself. It is hard to detect during ingest if a file is encrypted, but this may be known from the context in which it is ingested. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.

Usage notes

Several of the semantic components of signatureInformation are taken from the W3C's XML-Signature Syntax and Processing see http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/ for more information on the structure and application of these semantic units. (See also the discussion of digital signatures)

Table of Contents

1.8.1.1 signatureEncoding

Semantic components

None

Definition

The encoding used for the values of **signatureValue**, **keyInformation.**

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		Base64 Ds:CrytoBinary	Base64 Ds:CrytoBinary

Rationale

The values of **signatureValue** and keyInformation cannot be interpreted correctly if the encoding is unknown.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/signatureEncoding.html.

Creation/Maintenance notes

Inhibitors are more likely to be present on an object ingested by the repository than applied by the repository itself. It is hard to detect during ingest if a file is encrypted, but this may be known from the context in which it is ingested. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.

Table of Contents

1.8.1.2 signer

Semantic components

None

Definition

The individual, institution, or authority responsible for generating the signature.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional

Rationale

The signer might also be carried in the **keyInformation**, but it can be accessed more conveniently if recorded here.

Data Constraint

None

Usage notes

If the signer is an Agent known to the repository, an **agentIdentifier** can be used here.

Table of Contents

1.8.1.3 signatureMethod

Semantic components

None

Definition

A designation for the encryption and hash algorithms used for signature generation.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		DSA-SHA1 RSA-SHA1	DSA-SHA1 RSA-SHA1

Rationale

The same algorithms must be used for signature validation.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/signatureMethod.html.

Usage notes

Recommended practice is to use the following naming convention:name the encryption algorithm first, followed by a hyphen, followed by the name of the hash (message digest) algorithm.

Table of Contents

1.8.1.4 signatureValue

Semantic components

None

Definition

The digital signature; a value generated from the application of a private key to a message digest.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Examples		juS5RhJ884qoFR8flVXd/rbrSDVGn40CapgB7qeQiT+rr0NekEQ6BHhUA8dT3+BCTBUQI0dBjlml9lwzENXvS83zRECjzXbMRTUtVZiPZG2pqKPnL2YU3A9645UCjTXU+jgFumv7k78hieAGDzNci+PQ9KRmm//icT7JaYztgt4	juS5RhJ884qoFR8flVXd/rbrSDVGn40CapgB7qeQiT+rr0NekEQ6BHhUA8dT3+BCTBUQI0dBjlml9lwzENXvS83zRECjzXbMRTUtVZiPZG2pqKPnL2YU3A9645UCjTXU+jgFumv7k78hieAGDzNci+PQ9KRmm//icT7JaYztgt4

Data Constraint

None

Creation/Maintenance notes

Inhibitors are more likely to be present on an object ingested by the repository than applied by the repository itself. It is hard to detect during ingest if a file is encrypted, but this may be known from the context in which it is ingested. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.

Table of Contents

1.8.1.5 signatureValidationRules

Semantic components

None

Definition

The operations to be performed in order to validate the digital signature.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory

Rationale

The repository should not assume that the procedure for validating any particular signature will be known many years in the future without documentation.

Data Constraint

None

Creation/Maintenance notes

Inhibitors are more likely to be present on an object ingested by the repository than applied by the repository itself. It is hard to detect during ingest if a file is encrypted, but this may be known from the context in which it is ingested. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.

Usage notes

This may include the canonicalization method used before calculating the message digest, if the object was normalized before signing. This value could also be a pointer to archive documentation.

Table of Contents

1.8.1.6 signatureProperties

Semantic components

None

Definition

Additional information about the generation of the signature.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Data Constraint

None

Usage notes

This may include the date/time of signature generation, the serial number of the cryptographic hardware used, or other information related to the generation of the signature. Repositories will likely want to define a suitably granular structure to **signatureProperties.**

Table of Contents

1.8.1.7 keyInformation

Semantic components

Extensible container

Definition

Information about the signer’s public key needed to validate the digital signature.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional

Rationale

To validate a digital signature for an object, one first recalculates the message digest for the object, and then uses the public key of the signer to verify that the value of the signature (**signatureValue**) is correct. The repository must therefore have the public key value and some assurance that it truly belongs to the signer.

Data Constraint

None

Creation/Maintenance notes

Inhibitors are more likely to be present on an object ingested by the repository than applied by the repository itself. It is hard to detect during ingest if a file is encrypted, but this may be known from the context in which it is ingested. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.

Usage notes

Different types of keys will have different structures and parameters. PREMIS does not define the structure of this container. Recommended practice is to represent key values as defined for “KeyInfo” in the W3C’s XML-Signature Syntax and Processing (http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/).

Table of Contents

1.8.2 signatureInformationExtension

Semantic components

Defined externally

Definition

Digital signature information using semantic units defined outside of PREMIS.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional

Rationale

There may be a need to replace or extend PREMIS defined units.

Data Constraint

Container

Usage notes

For more granularity or use of externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata scheme may be included instead of or in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema must be provided. See further guidance in “Extensibility”.
If the **signatureInformationExtension** container needs to be associated explicitly with any PREMIS subunit under **signatureInformation**, the container **signatureInformation** is repeated. Also, if extensions from different external schemas are needed, **signatureInformation** should be repeated.Use of the W3C’s XML-Signature Syntax and Processing (http://

www.w3.org/TR/2002/REC-xmldsig-core-20020212/) is encouraged when applicable. It is recommended to give information about the metadata used in **signatureInformationExtension** including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

1.9 environmentFunction

Semantic components

1.9.1 environmentFunctionType
1.9.2 environmentFunctionLevel

Definition

A hierarchical description of the function of the environment used to render or execute an object.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Repeatable			
Obligation	Optional			

Rationale

This information describes the inherent nature of the environment which, in turn, specifies its function. It will not change over time and is not dependent on the way the environment is used. Types, such as software, documentation, hardware, must be described through an extensible vocabulary. It inherits from the previous PREMIS environment **software** and **hardware** semantic unit containers that specified fine-grained **swType** and **hwType** vocabulary. It is important to capture the type and use of an environment to express:

What is the type of the environment, such as hardware, software or documentation.
A refinement of the type to isolate where in the rendering stack the Intellectual Entity is used, such as plugin, driver, application, peripheral or specification.

Data Constraint

Container

Usage notes

environmentFunction can be used to describe the nature of an environment (whether it is hardware, software or documentation). This can be done on multiple, increasingly specific, hierarchical levels until the appropriate level of granularity has been reached. Multiple, hierarchical levels of description should be described in separate, repeated **environmentFunction** semantic unit containers. This can also be done by recording only the most specific level of granularity. For example, an environment Intellectual Entity describing a particular version of an operating system is also of the more generic type “software;” thus two levels of description (“software”, then “operating system”) describe the function. If it proved necessary, a third-level description could be added. There is no limit to the number of levels, although it is unlikely that a deep hierarchy will be necessary.

Table of Contents

1.9.1 environmentFunctionType

Semantic components

None

Definition

A description of the environment at a given level within the environmental stack.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Not repeatable			
Obligation	Mandatory			
Examples	**Level 1 types:** software hardware **Level 2 types:** software application software library software driver operating system plugin hardware architecture hardware peripheral chip			

Data Constraint

Container

Usage notes

The type values used at the higher levels are more generic and those used at the lower levels are more specific.

Table of Contents

1.9.2 environmentFunctionLevel

Semantic components

None

Definition

Level of the environment within an environmental stack.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Not repeatable			
Obligation	Optional if there is only one level; otherwise mandatory.			
Examples	1, 2			

Rationale

Distinguishes hierarchical levels of an environmental stack.

Data Constraint

Positive integer

Table of Contents

1.10 environmentDesignation

Semantic components

Definition

An identification of the environment used to render or execute an object.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Repeatable			
Obligation	Optional			

Rationale

Information identifying the environment using human-readable language which can be expected to be understood outside of a digital repository. This information is expected to be distinct from the **objectIdentifier** which uses a formal identification scheme to identify an object within the context of a repository and may not be readily understood outside of that context.

Data Constraint

Container

Table of Contents

1.10.1 environmentName

Semantic components

None

Definition

A commonly accepted name used to describe the environment.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Not repeatable			
Obligation	Mandatory			
Examples	Ubuntu			

Data Constraint

It is recommended to take this information from a controlled vocabulary.

Table of Contents

1.10.2 environmentVersion

Semantic components

None

Definition

Version of the environment.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Not repeatable			
Obligation	Mandatory			
Examples	Version 12.04			

Data Constraint

It is recommended to take this information from a controlled vocabulary.

Usage notes

Table of Contents

1.10.3 environmentOrigin

Semantic components

None

Definition

The origin of the environment referenced in **environmentName.**

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Not repeatable			
Obligation	Optional			
Examples	Microsoft The Document Foundation IBM			

Rationale

Software or hardware objects are products of an organization or an individual. This information is useful for preservation to further specify which software / hardware object is being described.

Data Constraint

Value should be taken from a controlled vocabulary.

Usage notes

Table of Contents

1.10.4 environmentDesignationNote

Semantic components

None

Definition

Any further information required to enhance the correct specification of the environment.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Repeatable			
Obligation	Optional			
Examples	32 bit			

Rationale

This semantic unit allows further information to be added in an unstructured way. This information could include restrictions, known issues, issues fixed in this release, information on execution time, etc.

Data Constraint

Value should be taken from a controlled vocabulary.

Usage notes

The Data Dictionary does not prescribe when to use a designation note and when to include the note in the name or version. For example, it is possible to describe the “Ubuntu 12.04 32 bit” operating system in a number of ways including:

Name = “Ubuntu”, Version = “12.04”, Designation Note = “32 bit
Name = “Ubuntu 32 bit” Version = “12.04”

Name = “Ubuntu” Version = “12.04 32-bit”
Name = “Ubuntu 12.04 32 bit”
All of these are acceptable, and which is used will depend on naming conventions in use within the repository, common usage by the individual or organization producing the environment, and other factors.

Table of Contents

1.10.5 environmentDesignationExtension

Semantic components

None

Definition

A container for describing the designation of an environment using semantic units defined outside of PREMIS.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Repeatable			
Obligation	Optional			

Rationale

There may be a need to supplement or replace PREMIS defined units.

Data Constraint

None

Usage notes

For more granularity or use of externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata scheme may be included in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema must be provided. See further guidance in “Extensibility”.
If **environmentDesignationExtension** needs to be associated explicitly with any PREMIS subunit under environmentDesignation, the container **environmentDesignation** is repeated. Also, if extensions from different

external schemas are needed, environmentDesignation should be repeated. **environmentDesignationExtension** is used for additional information about the environment designation beyond name and version in a structured way. This can be either additional information (such as release date, support dates for the software, manufacturing period for hardware etc.) or a finer-grained expression of version information (such as build number, serial number or installation guidelines). It is recommended to give information about the metadata used in **environmentDesignationExtension** including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

1.11 environmentRegistry

Semantic components

Definition

Identifies details about the registry where further information about the environment can be found.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Repeatable			
Obligation	Optional			

Rationale

If external registries are available to the preservation repository, they may provide an excellent way of referencing detailed information about the environment.

Data Constraint

Container

Table of Contents

1.11.1 environmentRegistryName

Semantic components

None

Definition

A designation identifying the referenced external registry.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable			
Repeatability	Not repeatable			
Obligation	Mandatory			
Examples	PRONOM			

Data Constraint

None

Table of Contents

1.11.2 environmentRegistryKey

Semantic components

None

Definition

The unique key used to reference an entry for this environment in an external registry.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Not repeatable			
Obligation	Mandatory			
Examples	sfw/2 x-sfw/2			

Data Constraint

None

Table of Contents

1.11.3 environmentRegistryRole

Semantic components

None

Definition

The purpose or expected use of the external registry.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Not repeatable			
Obligation	Optional			
Examples	Identification Specification			

Rationale

Registries could be used for different purposes or may not provide an identical match to the Intellectual Entity described in PREMIS. For example, one registry may only provide skeleton records that are sufficient to allow an environment to be formally identified (or might only describe a family of environments while the PREMIS object describes a more specific version within that family). An alternative registry may provide an exact match and provide much more detailed information needed to actually use the environment.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/environmentRegistryRole.

Table of Contents

1.12 environmentExtension

Semantic components

None

Definition

A container to include semantic units defined outside of PREMIS.

Entity information

	Intellectual Entity	Representation	File	Bitstream
Applicability	Applicable	Not applicable	Not applicable	Not applicable
Repeatability	Repeatable			
Obligation	Optional			

Rationale

There may be a need to extend PREMIS defined units.

Data Constraint

None

Usage notes

For more granularity or use of externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata scheme may be included in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema must be provided. See further guidance in “Extensibility”. If the extension is about the name, version or overall designation of the environment, use **environmentDesignationExtension. environmentExtension** is used for additional **environmentcharacteristics** not covered by PREMIS, for instance environment-specific metadata that is defined externally. It is not a replacement for units specified in PREMIS.If multiple extensions are needed (e.g., for different purposes), the container **environmentExtension**

is repeated. Also, if extensions from different external schemas are needed, **environmentExtension** should be repeated.It is recommended to give information about the metadata used in **environmentExtension** including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

1.13 relationship

Semantic components

Definition

Information about a relationship between this Object and one or more other Objects.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional

Rationale

A preservation repository must know how to assemble complex objects from component parts (structural relationships), rigorously track digital provenance (derivation relationships) and document the links between parts of a rendering stack in a Representation Information Network (dependency and documentation links). Documentation about relationships between different objects is crucial to these purposes.

Data Constraint

Container

Usage notes

Most preservation repositories will want to record all relevant relationships.In complex scenarios, PREMIS might not be able to express rich enough structural relationships to be the only source of structural metadata. Many formats for representing structural information may be used instead of the semantic units specified here. This information must be recorded, and some implementations may record it by using other structures (e.g. METS). Structural relationships between the fileand representation level are necessary to reconstruct a representation in order to ascertain that the representation is renderable. A record of structural relationships at the representation level may be necessary to render the representation. Structural relationships at the bitstream level can relate bitstreams within a file. Structural relationships between Intellectual Entities may be used to record logical containment, such as between an article and an issue, or physical containment, such as between a page and a book. Derivative relationships at the file, representation and Intellectual Entity level are important for documenting digital provenance.

Table of Contents

1.13.1 relationshipType

Semantic components

None

Definition

A high-level categorization of the nature of the relationship.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	derivation structural	derivation structural	derivation structural

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/relationshipType.html.

Usage notes

It is recommended to further categorize the relationship using **relationshipSubType.**

Table of Contents

1.13.2 relationshipSubType

Semantic components

None

Definition

A specific characterization of the nature of the relationship documented in relationshipType.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	Has root	Is source of Has part	Has source Is part of

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/relationshipSubType.html.

Usage notes

A repository may find it necessary to define more or less granular relationships than those values in the available controlled vocabulary. For derivation relationships, note that the precise relationship may be indicated by the type of the related Event. The relationship “has root” is applicable only to the representation, because it implies that a compound object (i.e. one made up of multiple files) requires that one file be picked up first as its root to render it. In the metadata for the representation, “has root” identifies that particular file.

Table of Contents

1.13.3 relatedObjectIdentifier

Semantic components

1.13.3.1 relatedObjectIdentifierType
1.13.3.2 relatedObjectIdentifierValue
1.13.3.3 relatedObjectSequence

Definition

The identifier and sequential context of the related Object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Mandatory	Mandatory	Mandatory

Data Constraint

Container

Usage notes

The related Object may or may not be held within the preservation repository. Recommended practice is that objects reside within the repository unless there is a good reason to reference an external Object. Internal and external references should be unambiguous.

Table of Contents

1.13.3.1 relatedObjectIdentifierType

Semantic components

None

Definition

A designation of the domain within which the identifier is unique.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	[see examples for **objectIdentifierType**]	[see examples for **objectIdentifierType**]	[see examples for **objectIdentifierType**]

Data Constraint

Value should be taken from a controlled vocabulary.

Usage notes

If the related Object is held within the preservation repository, this semantic unit should set to the value of that Object's **objectIdentifierType.**

Table of Contents

1.13.3.2 relatedObjectIdentifierValue

Semantic components

None

Definition

The value of the related Object identifier.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	[see examples for objectIdentifierValue]	[see examples for objectIdentifierValue]	[see examples for objectIdentifierValue]

Data Constraint

None

Usage notes

If the related Object is held within the preservation repository, this semantic unit should set to the value of that Object's **objectIdentifierValue.**

Table of Contents

1.13.3.3 relatedObjectSequence

Semantic components

None

Definition

The order of the related Object relative to other Objects with the same type of relationship.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Examples	1 2 3	1 2 3	1 2 3

Rationale

This semantic unit is particularly useful for structural relationships. In order to reconstruct a representation, it may be necessary to know the order of components with sibling or part-whole relationships. For example, to render the pages of a book, it is necessary to know the order of files representing pages.

Data Constraint

None

Usage notes

This semantic unit could be implemented in several ways. It might be recorded explicitly in metadata as a sequence number or as a pointer. It might be

implicit in some other ordering of Objects, for example, incrementing identifier values. The value of relationshipSubType might imply the sequence (e.g., “is preceding sibling,” “is following sibling”). There is no requirement that sequence numbers must be unique or sequential. Some related Objects have no inherent sequence, for example, unordered Web pages making up a website. It is acceptable to either use the “dummy” sequence number zero, or to omit completely. This semantic unit is applicable only for structural relationships and is thus optional.

Table of Contents

1.13.4 relatedEventIdentifier

Semantic components

1.13.4.1 relatedEventIdentifierType
1.13.4.2 relatedEventIdentifierValue
1.13.4.3 relatedEventSequence

Definition

The identifier and contextual sequence of an Event associated with the relationship.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional

Rationale

An Object may be related to another Object because of an Event, for example, migration.

Data Constraint

Container

Usage notes

For derivative relationships between Objects **relatedEventIdentifier** must be recorded.

Table of Contents

1.13.4.1 relatedEventIdentifierType

Semantic components

None

Definition

The **eventIdentifierType** of the related Event.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	[see examples for **eventIdentifierType**]	[see examples for **eventIdentifierType**]	[see examples for **eventIdentifierType**]

Data Constraint

Must be an existing **eventIdentifierType** value.

Usage notes

For most preservation repositories, the **eventIdentifierType** will simply be its own internal numbering system. It can be implicit within the system and provided explicitly only if the data is exported.

Table of Contents

1.13.4.2 relatedEventIdentifierValue

Semantic components

None

Definition

The **eventIdentifierValue** of the related Event.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	[see examples for **objectIdentifierValue**]	[see examples for **objectIdentifierValue**]	[see examples for **objectIdentifierValue**]

Data Constraint

Must be an existing eventIdentifierValue value.

Table of Contents

1.13.4.3 relatedEventSequence

Semantic components

None

Definition

The order of the related Event.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Examples	1 2 3	1 2 3	1 2 3

Data Constraint

Container

Usage notes

The sequence of a related Event can be inferred from the **eventDateTime** associated with the related Event.

Table of Contents

1.13.5 relatedEnvironmentPurpose

Semantic components

None

Definition

The use(s) supported by the related environment.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Examples	render edit create	render edit create	render edit create

Rationale

Different environments can support different uses of objects. For example, the environment needed to edit and modify a file can be quite different from the environment needed to render it.

Data Constraint

Values should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/environmentPurpose.

Creation/Maintenance notes

This value would have to be supplied by the Agent that provided the hardware and/or software environment information, which might be the submitter, the repository, or an environment registry.

Usage notes

Other values might indicate the ability to transform, print, and manipulate by program.

Table of Contents

1.13.6 relatedEnvironmentCharacteristic

Semantic components

None

Definition

An assessment of the extent to which the described environment supports its purpose.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Examples	**For Representation:** unspecified minimum known to work recommended	unspecified minimum unspecified minimum	unspecified minimum unspecified minimum

Rationale

If multiple environments are described, this element can help to distinguish between them.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/environmentCharacteristic.

Creation/Maintenance notes

This value could be supplied by the submitter or by the repository. If environment software and hardware information is obtained from an environment registry, relatedEnvironmentCharacteristic might also be obtained from the registry. Note however that the criteria for “recommended” may be different for different repositories.

Usage notes

If an environment could be described as both “minimum” and “recommended,” use “recommended.” “Known to work” implies the object is supported by the described environment but the repository doesn’t know if this environment is minimum or recommended.

Table of Contents

1.14 linkingEventIdentifier

Semantic components

1.14.1 linkingEventIdentifierType
1.14.2 linkingEventIdentifierValue

Definition

The **eventIdentifierType** value of the related Event.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional

Data Constraint

Container

Usage notes

Use to link to Events that are not associated with relationships between Objects, such as format validation, virus checking, etc. Linking semantic units are mandatory in the sense that a repository needs to know the information, but are defined as optional because PREMIS does not specify in which direction the linkage should be.

Table of Contents

1.14.1 linkingEventIdentifierType

Semantic components

None

Definition

The **eventIdentifierType** value of the related Event.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	[see examples for **eventIdentifierType**]	[see examples for **eventIdentifierType**]	[see examples for **eventIdentifierType**]

Data Constraint

Must be an existing **eventIdentifierType** value.

Table of Contents

1.14.2 linkingEventIdentifierValue

Semantic components

None

Definition

The eventIdentifierValue value of the related Event.

Entity information

	Intellectual Entity / Representation	**File**	**Bitstream**
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	[see examples for **eventIdentifierValue**]	[see examples for **eventIdentifierValue**]	[see examples for **eventIdentifierValue**]

Data Constraint

Must be an existing **eventIdentifierValue** value.

Table of Contents

1.15 linkingRightsStatementIdentifier

Semantic components

Definition

An identifier for a Rights statement associated with the object.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional

Rationale

A repository may choose to link from a Rights statement to an object or from an object to a Rights statement or both.

Data Constraint

Container

Usage notes

Linking semantic units are mandatory in the sense that a repository needs to know the information, but are defined as optional because PREMIS does not specify in which direction the linkage should be.

Table of Contents

1.15.1 linkingRightsStatementIdentifierType

Semantic components

None

Definition

A designation of the domain within which the **linkingRightsStatementIdentifier** is unique.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Examples	URI LCCN	URI LCCN	URI LCCN

Data Constraint

Must be an existing **rightsStatementIdentifierType** value.

Table of Contents

1.15.2 linkingRightsStatementIdentifierValue

Semantic components

None

Definition

A designation of the domain within which the linkingRightsStatementIdentifier is unique.

Entity information

	Intellectual Entity / Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

Data Constraint

Must be an existing **rightsStatementIdentifierType** value.

Table of Contents

Events

Event Entity

- The Event entity aggregates information about an action that involves one or more Object entities. Metadata about an Event would normally be recorded and stored separately from the digital object.
- Whether or not a preservation repository records an Event depends upon the importance of the event. Actions that modify objects should always be recorded. Other actions such as copying an object for backup purposes may be recorded in system logs or an audit trail but not necessarily in an Event entity
- Mandatory semantic units are: eventIdentifier, eventType, and eventDateTime.

Entity properties

- Must be related to one or more Objects.
- Can be related to one or more Agents.
- Links between entities may be recorded from either direction and need not be bi-directional.

Entity semantic units

2.1 eventIdentifier (M, NR)
  2.1.1 eventIdentifierType (M, NR)
  2.1.2 eventIdentifierValue (M, NR)
2.2 eventType (M, NR)
2.3 eventDateTime (M, NR)
2.4 eventDetailInformation (O, R)
  2.4.1 eventDetail (O, NR)
  2.4.2 eventDetailExtension (O, R)
2.5 eventOutcomeInformation (O, R)
  2.5.1 eventOutcome (O, NR)
  2.5.2 eventOutcomeDetail (O, R)
    2.5.2.1 eventOutcomeDetailNote (O, NR)
    2.5.2.2 eventOutcomeDetailExtension (O, R)
2.6 linkingAgentIdentifier (O, R)
  2.6.1 linkingAgentIdentifierType (M, NR)
  2.6.2 linkingAgentIdentifierValue (M, NR)
  2.6.3 linkingAgentRole (O, R)
2.7 linkingObjectIdentifier (O, R)
  2.7.1 linkingObjectIdentifierType (M, NR)
  2.7.2 linkingObjectIdentifierValue (M, NR)
  2.7.3 linkingObjectRole (O, R)

2.1 eventIdentifier

Semantic components

Definition

A designation used to identify the Event uniquely within the preservation repository system.

Entity information

Obligation	Mandatory
Repeatability	Not Repeatable

Rationale

Each Event recorded by the preservation repository must have a unique identifier to allow it to be related to Objects, Agents, and other Events.

Data Constraint

Container

Example

Creation/Maintenance notes

The **eventIdentifier** is likely to be system generated. There is no global scheme or standard for these identifiers. The identifier is therefore not repeatable.

Usage notes

Table of Contents

2.1.1 eventIdentifierType

Semantic components

None

Definition

A designation of the domain within which the Event identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Not Repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary.

Example

FDA
Stanford Repository Event I
UUID
local

Creation/Maintenance notes

For most preservation repositories, the **eventIdentifierType** will be its own internal numbering system. It can be implicit within the system and provided explicitly only if the data is exported.

Usage notes

Table of Contents

2.1.2 eventIdentifierValue

Semantic components

None

Definition

The value of the **eventIdentifier.**

Entity information

Obligation	Mandatory
Repeatability	Not Repeatable

Rationale

Data Constraint

None

Example

[a binary integer]
E-2004-11-13-000119
58f202ac-22cf-11d1-b12d-002035b2909

Creation/Maintenance notes

Usage notes

Table of Contents

2.2 eventType

Semantic components

None

Definition

A categorization of the nature of the Event.

Entity information

Obligation	Mandatory
Repeatability	Not Repeatable

Rationale

Categorizing Events will aid the preservation repository in machine processing of Event information, particularly in reporting.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/eventType.

Example

E77 [a code used within a repository for a particular Event type]
ingestion
migration
validation

Creation/Maintenance notes

Usage notes

Some **eventType** values may be more generic than others: migration, normalization, and replication are in some instances more precise subtypes of the creation **eventType** value. “Creation” can be used when more precise terms do not apply, for example, when a digital object was first created by scanning from paper. In general, the level of specificity in recording the type of Event (e.g., whether the **eventType** indicates a transformation, a migration or a particular method of migration) is implementation specific and will depend upon how reporting and processing is done. Recommended practice is to record detailed information about the Event itself in **eventDetailInformation** rather than using a very granular value for **eventType**.

Table of Contents

2.3 eventDateTime

Semantic components

None

Definition

The single date and time, or date and time range, at or during which the Event occurred.

Entity information

Obligation	Mandatory
Repeatability	Notrepeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

20050704T071530-0500 [July 4, 2005 at 7:15:30 a.m. EST]
2006-07-16T19:20:30+01:00
20050705T0715-0500/20050705T0720-0500 [from 7:15 a.m. EST to 7:20 a.m. EST on July 4, 2005]
2004-03-17 [March 17, 2004, only the date is known]

Creation/Maintenance notes

Usage notes

Recommended practice is to record the most specific time possible and to designate the time zone.

Table of Contents

2.4 eventDetailInformation

Semantic components

2.4.1 eventDetail
2.4.2 eventDetailExtension

Definition

Additional information about the Event.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

All subunits of this semantic unit are optional but at least one subunit (i.e. **eventDetail** and/or **eventDetailExtension**) must be present if this container is included.

Table of Contents

2.4.1 eventDetail

Semantic components

None

Definition

Additional information about the Event.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

None

Example

Object permanently withdrawn by request of Caroline Hunt.
Program = “MIGJP2JP2K”; version = “2.2”

Creation/Maintenance notes

Usage notes

eventDetail is not necessarily intended to be processed by machine. It may record any information about an Event and/or point to information stored elsewhere. When multiple details about the same Event need to be recorded, the **eventDetailInformation** container must be repeated. **eventDetailExtension** allows more expressivity if required.

Table of Contents

2.4.2 eventDetailExtension

Semantic components

Defined externally

Definition

A container to include semantic units defined outside of PREMIS.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

There may be a need to replace or extend PREMIS defined units.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

For more granularity or use of externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata schema may be included instead of or in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema must be provided. See further guidance in “Extensibility”, . If more than one extension needs to be associated explicitly with **eventDetail**,

eventDetailExtension is repeated. However, if extensions from different external schemas are needed or if the extension is not associated explicitly with **eventDetail**, the **eventDetailInformation** container should be repeated instead. It is recommended to give information about the metadata used in **eventDetailExtension**, including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

2.5 eventOutcomeInformation

Semantic components

Definition

Information about the outcome of an Event.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

A repository may wish to supplement a coded **eventOutcome** value with additional information in **eventOutcomeDetail**. Since Events may have more than one outcome, the container is repeatable. All subunits of this semantic unit are optional. At least one subunit (i.e. **eventOutcome** or **eventOutcomeDetail**) must be present if this container is included.

Table of Contents

2.5.1 eventOutcome

Semantic components

None

Definition

A categorization of the overall result of the Event in terms of success, partial success, or failure.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

A coded way of representing the outcome of an Event may be useful for machine processing and reporting. If, for example, a fixity check fails, the Event record provides both an actionable and a permanent record.

Data Constraint

Value should be taken from a controlled vocabulary.

Example

00 [a code meaning “action successfully completed”]
CV-01 [a code meaning “checksum validated”]

Creation/Maintenance notes

Usage notes

Recommended practice is to use a controlled vocabulary that a system can act upon automatically. Because this is inherently a local vocabulary, there is no general one available. More detail about the outcome may be recorded in **eventOutcomeDetail**. Recommended practice is to define Events with sufficient granularity that each Event has a single outcome.

Table of Contents

2.5.2 eventOutcomeDetail

Semantic components

Definition

A detailed description of the result or product of the Event.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

An Event outcome may be sufficiently complex that a coded description is not adequate to document it.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

This may be used to record all error and warning messages issued by a program involved in the Event or to record a pointer to an error log. If the Event was a validity check (e.g., profile conformance) any anomalies or quirks discovered would be recorded here. All subunits of this semantic unit

are optional. At least one subunit (i.e. **eventOutcomeDetailNote** and/or **eventOutcomeDetailExtension**) must be present if this container is included.

Table of Contents

2.5.2.1 eventOutcomeDetailNote

Semantic components

None

Definition

A detailed description of the result or product of the Event in textual form.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Additional information in textual form may be needed about the outcome of the Event.

Data Constraint

None

Example

LZW compressed file
Non-standard tags found in header

Creation/Maintenance notes

Usage notes

Table of Contents

2.5.2.2 eventOutcomeDetailExtension

Semantic components

Defined externally

Definition

A container to include semantic units defined outside of PREMIS.

Entity information

Optional
Repeatable

Rationale

There may be a need to replace or extend PREMIS defined units.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

For more granularity or use of externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata schema may be included instead of or in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema

must be provided. See further guidance in “Extensibility,” . If more than one extension needs to be associated explicitly with **eventOutcomeDetailNote**, **eventOutcomeDetailExtension** is repeated. However, if extensions from different external schemas are needed or if the extension is not associated explicitly with **eventOutcomeDetailNote**, the **eventOutcomeDetail** container should be repeated instead. It is recommended to give information about the metadata used in **eventOutcomeDetailExtension** including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

2.6 linkingAgentIdentifier

Semantic components

2.6.1 linkingAgentIdentifierType
2.6.2 linkingAgentIdentifierValue
2.6.3 linkingAgentRole

Definition

Identification of one or more Agents associated with the Event.

Entity information

Optional
Repeatable

Rationale

Digital provenance often requires that relationships between Agents and Events are documented.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Recommended practice is to record the Agent if possible.Linking semantic units are mandatory in the sense that a repository needs to know the information,

but are defined as optional because PREMIS does not specify in which direction the linkage should be.

Table of Contents

2.6.1 linkingAgentIdentifierType

Semantic components

None

Definition

A designation of the domain in which the linking Agent identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Data Constraint

Must be an existing agentIdentifierType value.

Example

[see examples for agentIdentifierType

Creation/Maintenance notes

Usage notes

Table of Contents

2.6.2 linkingAgentIdentifierValue

Semantic components

None

Definition

The value of the linking Agent identifier.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Data Constraint

Must be an existing **agentIdentifierValue** value.

Example

[see examples for agentIdentifierValue]

Creation/Maintenance notes

Usage notes

Table of Contents

2.6.3 linkingAgentRole

Semantic components

None

Definition

The role of the Agent in relation to this Event.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Events can have more than one Agent associated with them. The role of each Agent may need to be documented.

Data Constraint

Values should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/eventRelatedAgentRole.

Example

Creation/Maintenance notes

authorizer

implementer

validator

executing program

Usage notes

Table of Contents

2.7 linkingObjectIdentifier

Semantic components

2.7.1 linkingObjectIdentifierType
2.7.2 linkingObjectIdentifierValue
2.7.3 linkingObjectRole

Definition

Information about an Object associated with an Event.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Digital provenance often requires that relationships between Objects and Events are documented.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Linking semantic units are mandatory in the sense that a repository needs to know the information, but are defined as optional because PREMIS does not specify in which direction the linkage should be.

Table of Contents

2.7.1 linkingObjectIdentifierType

Semantic components

None

Definition

A designation of the domain in which the linking Object identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Data Constraint

Must be an existing **objectIdentifierType** value.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

2.7.2 linkingObjectIdentifierValue

Semantic components

None

Definition

The value of the linking Object identifier.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Data Constraint

Must be an existing **objectIdentifierValue** value.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

2.7.3 linkingObjectRole

Semantic components

None

Definition

The role of the Object associated with an Event.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Distinguishes the role of the Object in relation to an Event. If this is not explicit it is necessary to analyze the relationship between Objects in the Object metadata.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/eventRelatedObjectRole.

Example

source
outcome

Creation/Maintenance notes

Usage notes

Table of Contents

Agents

Agent Entity

The Agent entity aggregates information about attributes or characteristics of Agents (persons, organizations, or software) associated with Rights management and preservation events in the life of a data object. Agent information serves to identify an Agent unambiguously from all other Agent entities.

A piece of software or hardware that is captured as an Agent can also be preserved in a repository and described as an environment O bject, for example, as source code or an ISO disk image. In this case, implementers may choose to relate the Agent to the environment O bject using the linkingEnvironmentIdentifier semantic unit. This relationship can support the ability of a repository to record in considerable detail the interactions between software or hardware Agents and the preserved digital objects with which they interact, and even to reproduce these interactions if desired.

The only mandatory semantic unit is agentIdentifier.

Entity properties

- May hold or grant one or more Rights.
- May carry out, authorize, or compel one or more Events.
- May create or act upon one or more Objects through an Event or with respect to a Rights statement.

Entity semantic units

3.1 agentIdentifier (M, R)
  3.1.1 agentIdentifierType (M, NR)
  3.1.2 agentIdentifierValue (M, NR)
3.2 agentName (O, R)
3.3 agentType (O, NR)
3.4 agentVersion (O, NR)
3.5 agentNote (O, R)
3.6 agentExtension (O, R)
3.7 linkingEventIdentifier (O, R)
  3.7.1 linkingEventIdentifierType (M, NR)
  3.7.2 linkingEventIdentifierValue (M, NR)
3.8 linkingRightsStatementIdentifier (O, R)
  3.8.1 linkingRightsStatementIdentifierType (M, NR)
  3.8.2 linkingRightsStatementIdentifierValue (M, NR)
3.9 linkingEnvironmentIdentifier (O, R)
  3.9.1 linkingEnvironmentIdentifierType (M, NR)
  3.9.2 linkingEnvironmentIdentifierValue (M, NR)

3.9.3 linkingEnvironmentRole (O, R)

3.1 agentIdentifier

Semantic components

3.1.1 agentIdentifierType
3.1.2 agentIdentifierValue

Definition

The designation used to uniquely identify the Agent within a preservation repository system.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Each Agent associated with the preservation repository must have a unique identifier to allow it to be related to Events and Rights statements.

Data Constraint

Container

Example

Creation/Maintenance notes

An identifier may be created by the repository system, or it may be created or assigned outside of the repository. Similarly, identifiers can be automatically or manually generated. Recommended practice is for repositories to use an identifier automatically created by the repository as the primary identifier in order to ensure that identifiers are unique and usable by the repository. Externally assigned identifiers can be used as secondary identifiers in order to link an Agent to information held outside the repository.

Usage notes

Identifiers must be unique within the repository. The **agentIdentifier** is repeatable in order to allow both repository-assigned and externally-assigned identifiers to be recorded. See Creation/Maintenance notes, above.

Table of Contents

3.1.1 agentIdentifierType

Semantic components

None

Definition

A designation of the domain in which the Agent identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary.

Example

LCNAF
SAN
DLC
URI
local

Creation/Maintenance notes

Usage notes

Table of Contents

3.1.2 agentIdentifierValue

Semantic components

None

Definition

The value of the agentIdentifier.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Data Constraint

Value may be taken from a controlled vocabulary. If **agentType** is organization, a controlled vocabulary is available at: http://id.loc.gov/vocabulary/organizations.

Example

92-79971
Owens, Erik C.
234-5676 MH-CSinfo:lccn/n78890351
MH-CSinfo:lccn/n7889035
info:lccn/n78890351

Creation/Maintenance notes

Usage notes

May be a unique key or a controlled textual form of name.

Table of Contents

3.2 agentName

Semantic components

None

Definition

A text string which could be used in addition to agentIdentifierto identify an Agent.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

None

Example

Erik
Erik Owens
Woodyard
JHOVE

Creation/Maintenance notes

Usage notes

The value is not necessarily unique. If **agentType** is software, **agentVersion** can be used to refine **agentName**.

Table of Contents

3.3 agentType

Semantic components

None

Definition

A high-level characterization of the type of Agent.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/agentType.html.

Example

person
organization
software
hardware

Creation/Maintenance notes

Usage notes

Table of Contents

3.4 agentVersion

Semantic components

None

Definition

The version of the Agent referenced in **agentName**, if **agentType** is software or hardware.

Entity information

Obligation	Optional
Repeatability	Not Repeatable

Rationale

Software or hardware Agents can behave very differently from one version to another.

Data Constraint

None

Example

1.6
2.1.0
20120521

Creation/Maintenance notes

Usage notes

This semantic unit only applies to machine Agents (hardware or software). If there is no formal version, the date of issuance may be used. The distinction between **agentName** and **agentVersion** can be fuzzy in some cases. Sometimes, a new version of a product becomes a standalone product of its own. For instance, even though JHOVE 2 declares itself as the second major version of the JHOVE file analysis software application, it is a very different product with a different architecture, features and maintaining agency; currently, JHOVE 1 and JHOVE 2 continue to evolve in parallel. For those reasons, it would be better to have JHOVE 2 in **agentName**, and record the specific build number of the software used in the repository in **agentVersion**. In any case, implementers should define their own naming policies to express name and version information and apply it consistently for all their Agent descriptions.If the Agent is also described as an environment Object, implementers may choose to link the Agent to the **environmentObject** using the **linkingEnvironmentIdentifier** semantic unit. In this case, the information in **agentVersion** and **environmentVersion** should be consistent.

Table of Contents

3.5 agentNote

Semantic components

None

Definition

Additional information about the Agent.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Additional information may be needed to describe or disambiguate the Agent.

Data Constraint

None

Example

--prefix = /opt/local
[configuration options used with a software Agent

Creation/Maintenance notes

Usage notes

Use **agentNote** when relatively limited information must be supplied. If extensive additional information is required, consider using an externally defined schema with **agentExtension** instead.

Table of Contents

3.6 agentExtension

Semantic components

Defined externally

Definition

A container to include semantic units defined outside of PREMIS.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

There may be a need to replace or extend PREMIS defined units.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

For more granularity or use of externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata scheme may be included instead of or in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema must be provided. See further guidance in “Extensibility,” INSERT LINK HERE. It is recommended to give information about the metadata used

in agentExtension including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

3.7 linkingEventIdentifier

Semantic components

3.7.1 linkingEventIdentifierType
3.7.2 linkingEventIdentifierValue

Definition

The **eventIdentifier** of an Event associated with the Agent.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Linking semantic units are mandatory in the sense that a repository needs to know the information, but are defined as optional because PREMIS does not specify in which direction the linkage should be.

Table of Contents

3.7.1 linkingEventIdentifierType

Semantic components

None

Definition

The **eventIdentifierType** value of the related Event.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Data Constraint

Must be an existing **eventIdentifierType** value.

Example

[see examples for eventIdentifierType]

Creation/Maintenance notes

Usage notes

Table of Contents

3.7.2 linkingEventIdentifierValue

Semantic components

None

Definition

The **eventIdentifierValue** value of the related Event.

Entity information

Obligation	Mandatory
Repeatability	Repeatable

Rationale

Data Constraint

Must be an existing **eventIdentifierValue** value.

Example

[see examples for eventIdentifierValue]

Creation/Maintenance notes

Usage notes

Table of Contents

3.8 linkingRightsStatementIdentifier

Semantic components

3.8.1 linkingRightsStatementIdentifierType
3.8.2 linkingRightsStatementIdentifierValue

Definition

An identifier for a Rights statement associated with the Agent.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

A repository may choose to link from a Rights statement to an Agent or from an Agent to a Rights statement or both.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Linking semantic units are mandatory in the sense that a repository needs to know the information, but are defined as optional because PREMIS does not specify in which direction the linkage should be.

Table of Contents

3.8.1 linkingRightsStatementIdentifierType

Semantic components

None

Definition

A designation of the domain within which the **linkingRightsStatementIdentifier** is unique.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Must be an existing **rightsStatementIdentifierType** value.

Example

URI
LCCN

Creation/Maintenance notes

Usage notes

Table of Contents

3.8.2 linkingRightsStatementIdentifierValue

Semantic components

None

Definition

The value of the **linkingRightsStatementIdentifier**.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Must be an existing **rightsStatementtIdentifierValue** value.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

3.9 linkingEnvironmentIdentifier

Semantic components

3.9.1 linkingEnvironmentIdentifierType
3.9.2 linkingEnvironmentIdentifierValue
3.9.3 linkingEnvironmentRole

Definition

The objectIdentifier of an environment Object associated with the Agent.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Allows implementers to link the Agent to an environment Object, if the Agent is hardware or software that has been preserved in some way in the repository and has been described as an environment Object.

Data Constraint

Container

Example

Allows implementers to link the Agent to an environment Object, if the Agent i s hardware or software that has been preserved in some way in the repository a nd has been described as an environment Object.

Creation/Maintenance notes

Usage notes

Although the semantic unit is named **linkingEnvironmentIdentifier**, it links an Agent to an Object. **LinkingEnvironmentIdentifierType** and **linkingEnvironmentIdentifierValue** must therefore match **objectIdentifierType** and **objectIdentifierValue** in the related environment Object.

Table of Contents

3.9.1 linkingEnvironmentIdentifierType

Semantic components

None

Definition

A designation of the domain within which the **linkingEnvironmentIdentifier** is unique.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Must be an existing **objectIdentifierType** value.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

3.9.2 linkingEnvironmentIdentifierValue

Semantic components

Definition

The value of the **linkingEnvironmentIdentifier**.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Must be an existing **objectIdentifierValue**.

Example

[see examples for objectIdentifierValue]

Creation/Maintenance notes

Usage notes

Table of Contents

3.9.3 linkingEnvironmentRole

Semantic components

None

Definition

The role of the environment Object associated with this Agent.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

A piece of software or hardware that is captured as an Agent can also be preserved in a repository and described as an environment Object: for example, as source code or an ISO disk image. The role allows one to record in what form the software or hardware has been preserved in the repository.

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/linkingEnvironmentRole.html.

Example

source code
bytecode
disk image

Creation/Maintenance notes

Usage notes

Table of Contents

Rights

Rights Entity

For the purpose of the PREMIS Data Dictionary, statements of legal rights and permissions are taken to be constructs that can be described as the Rights entity. Rights are entitlements grantedto Agents by copyright or other intellectual property law. Permissions are powers or privileges granted by agreement between a rightsholder and another party or parties.

A repository might wish to record a variety of Rights information including abstract Rightsstatements and statements of permissions that apply to external Agents and to objects not held within the repository. The minimum core Rights information that a preservation repository must know, however, is what Rights or permissions a repository has to carry out actions related to objects within the repository. These may generally be granted by copyright law, by statute, or by a license agreement with the rightsholder. In some situations the basis for the rights is for other reasons, for instance institutional policy.

If the repository records Rights information, either **rightsStatement** or **rightsExtension** must be present.

Entity properties

- May be related to one or more Objects.
- May be related to one or more Agents.
- Links between entities may be recorded from either direction and need not be bi-directional.

Entity semantic units

4.1 rightsStatement (O, R)
- 4.1.1 rightsStatementIdentifier (M, NR)
  - 4.1.1.1 rightsStatementIdentifierType (M, NR)
  - 4.1.1.2 rightsStatementIdentifierValue (M, NR)
- 4.1.2 rightsBasis (M, NR)
- 4.1.3 copyrightInformation (O, NR)
  - 4.1.3.1 copyrightStatus (M, NR)
  - 4.1.3.2 copyrightJurisdiction (M, NR)
  - 4.1.3.3 copyrightStatusDeterminationDate (O, NR)
  - 4.1.3.4 copyrightNote (O, R)
  - 4.1.3.5 copyrightDocumentationIdentifier (O, R)
    - 4.1.3.5.1 copyrightDocumentationIdentifierType (M, NR)
    - 4.1.3.5.2 copyrightDocumentationIdentifierValue (M, NR)
    - 4.1.3.5.3 copyrightDocumentationRole (O, NR)
  - 4.1.3.6 copyrightApplicableDates (O, NR)
    - 4.1.3.6.1 startDate (O, NR)

- 4.1.3.6.2 endDate (O, NR)
- 4.1.4 licenseInformation (O, NR)
  - 4.1.4.1 licenseDocumentationIdentifier (O, R)
    - 4.1.4.1.1 licenseDocumentationIdentifierType (M, NR)
    - 4.1.4.1.2 licenseDocumentationIdentifierValue (M, NR)
    - 4.1.4.1.3 licenseDocumentationRole (O, NR)
  - 4.1.4.2 licenseTerms (O, NR)
  - 4.1.4.3 licenseNote (O, R)
  - 4.1.4.4 licenseApplicableDates (O, NR)
    - 4.1.4.4.1 startDate (O, NR)
    - 4.1.4.4.2 endDate (O, NR)
- 4.1.5 statuteInformation (O, R)
  - 4.1.5.1 statuteJurisdiction (M, NR)
  - 4.1.5.2 statuteCitation (M, NR)
  - 4.1.5.3 statuteInformationDeterminationDate (O, NR)
  - 4.1.5.4 statuteNote (O, R)
  - 4.1.5.5 statuteDocumentationIdentifier (O, R)
    - 4.1.5.5.1 statuteDocumentationIdentifierType (M, NR)
    - 4.1.5.5.2 statuteDocumentationIdentifierValue (M, NR)
    - 4.1.5.5.3 statuteDocumentationRole (O, NR)
  - 4.1.5.6 statuteApplicableDates (O, NR)
    - 4.1.5.6.1 startDate (O, NR)
    - 4.1.5.6.2 endDate (O, NR)
- 4.1.6 otherRightsInformation (O, NR)
  - 4.1.6.1 otherRightsDocumentationIdentifier (O, R)
    - 4.1.6.1.1 otherRightsDocumentationIdentifierType (M, NR)
    - 4.1.6.1.2 otherRightsDocumentationIdentifierValue (M, NR)
    - 4.1.6.1.3 otherRightsDocumentationRole (O, NR)
  - 4.1.6.2 otherRightsBasis (M, NR)
  - 4.1.6.3 otherRightsApplicableDates (O, NR)
    - 4.1.6.3.1 startDate (O, NR)
    - 4.1.6.3.2 endDate (O, NR)
  - 4.1.6.4 otherRightsNote (O, R)
- 4.1.7 rightsGranted (O, R)
  - 4.1.7.1 act (M, NR)
  - 4.1.7.2 restriction (O, R)
  - 4.1.7.3 termOfGrant (O, NR)
    - 4.1.7.3.1 startDate (M, NR)
    - 4.1.7.3.2 endDate (O, NR)
  - 4.1.7.4 termOfRestriction (O, NR)
    - 4.1.7.4.1 startDate (M, NR)
    - 4.1.7.4.2 endDate (O, NR)
  - 4.1.7.5 rightsGrantedNote (O, R)
- 4.1.8 linkingObjectIdentifier (O, R)
  - 4.1.8.1 linkingObjectIdentifierType (M, NR)
  - 4.1.8.2 linkingObjectIdentifierValue (M, NR)
  - 4.1.8.3 linkingObjectRole (O, R)

4.1.9 linkingAgentIdentifier (O, R)
4.1.9.1 linkingAgentIdentifierType (M, NR)
4.1.9.2 linkingAgentIdentifierValue (M, NR)
4.1.9.3 linkingAgentRole (O, R)
4.2 rightsExtension (O, R)

4.1 rightsStatement

Semantic components

Definition

Documentation of the repository's Rights or indeed restrictions to perform one or more acts.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

This semantic unit is optional because in some cases Rights may be unknown. Institutions are encouraged to record Rights information when possible. Either **rightsStatement** or **rightsExtension** must be present if the Rights entity is included. The **rightsStatement** should be repeated when the act(s) described has(have) more than one basis, or when different acts have different bases.

Table of Contents

4.1.1 rightsStatementIdentifier

Semantic components

4.1.1.1 rightsStatementIdentifierType
4.1.1.2 rightsStatementIdentifierValue

Definition

The designation used to identify the Rights statement uniquely within a preservation repository system.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Each statement of Rights associated with the preservation repository must have a unique identifier to allow it to be related to Events and Agents.

Data Constraint

Container

Example

Creation/Maintenance notes

The **rightsStatementIdentifier** is likely to be system generated. There is no global scheme or standard for these identifiers. The identifier is therefore not repeatable.

Usage notes

Identifiers must be unique within the repository.

Table of Contents

4.1.1.1 rightsStatementIdentifierType

Semantic components

None

Definition

A designation of the domain within which the Rights statement identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary.A controlled vocabulary is available at: http://id.loc.gov/vocabulary/identifiers.html

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.1.2 rightsStatementIdentifierValue

Semantic components

None

Definition

The value of the rightsStatementIdentifier.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.2 rightsBasis

Semantic components

None

Definition

Designation of the basis for the right or permission identified by the **rightsStatementIdentifier**.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Example

copyright
license
statute
other

Creation/Maintenance notes

Usage notes

When **rightsBasis** is “copyright”, **copyrightInformation** should be provided.
When **rightsBasis** is “license”, **licenseInformation** should be provided.
When **rightsBasis** is “statute”, **statuteInformation** should be provided.

When **rightsBasis** is “other”, **otherRightsBasis** (in **otherRightsInformation** container) should be provided. If the Rights for the item are public domain, use “copyright”. If they are Fair Use, use “statute”. If more than one basis applies, the entire Rights entity should be repeated.

Table of Contents

4.1.3 copyrightInformation

Semantic components

Definition

Information about the copyright status of the object(s).

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

When **rightsBasis** is “copyright”, **copyrightInformation** should be provided. Repositories may need to extend this with more detailed information. See

the California Digital Library's copyrightMD schema (http://www.cdlib.org/groups/rmg//) for an example of a more detailed scheme.

Table of Contents

4.1.3.1 copyrightStatus

Semantic components

None

Definition

A coded designation for the copyright status of the object at the time the Rights statement was recorded.

Entity information

Obligation	Mandatory
Repeatability	Not epeatable

Rationale

Data Constraint

Values should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/copyrightStatus.html.

Example

copyrighted = Under copyright
publicdomain = In the public domain
unknown = Copyright status of the resource is unknown

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.3.2 copyrightJurisdiction

Semantic components

None

Definition

The country whose copyright laws apply.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Copyright law can vary from country to country.

Data Constraint

Values should be taken from ISO 3166.

Example

us
de

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.3.3 copyrightStatusDeterminationDate

Semantic components

None

Definition

The date that the copyright status recorded in **copyrightStatus** was determined.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

20070608

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.3.4 copyrightNote

Semantic components

None

Definition

Additional information about the copyright status of the object.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

None

Example

Copyright expiration expected in 2010 unless renewed.
Copyright statement is embedded in file header.

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.3.5 copyrightDocumentationIdentifier

Semantic components

4.1.3.5.1 copyrightDocumentationIdentifierType
4.1.3.5.2 copyrightDocumentationIdentifierValue
4.1.3.5.3 copyrightDocumentationRole

Definition

A designation used to identify documentation supporting the specified Rights granted according to copyright uniquely within the repository system.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

This semantic unit is intended to refer to a document detailing the granting of permission when the Rights basis is copyright. If repeated,

use **copyrightDocumentationRole** to distinguish the role of the given documentation.

Table of Contents

4.1.3.5.1 copyrightDocumentationIdentifierType

Semantic components

None

Definition

A designation of the domain within which the copyrightdocumentation identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/identifiers.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.3.5.2 copyrightDocumentationIdentifierValue

Semantic components

None

Definition

The value of the **copyrightDocumentationIdentifier**.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

None

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.3.5.3 copyrightDocumentationRole

Semantic components

None

Definition

A value indicating the purpose or expected use of the documentation being identified.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

This information distinguishes the purpose of the supporting documentation especially when there are multiple documentation identifiers.

Data Constraint

Value should be taken from a controlled vocabulary.

Example

accession record
copyright statement

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.3.6 copyrightApplicableDates

Semantic components

4.1.3.6.1 startDate
4.1.3.6.2 endDate

Definition

The date range during which the particular copyright applies or is applied to the content. This is distinct from **termOfGrant**, which applies to a particular act expressed in **rightsGranted** and may differ from the period of time the license, statute, or other basis applies to the content.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Specific dates may apply to the particular copyright granted.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

The repository may wish to retain the history of Rights and restrictions associated with the content over time. Associating active dates with particular

Rights bases allows applications to identify which of several **rightsStatements** are in force at a given time.

Table of Contents

4.1.3.6.1 startDate

Semantic components

None

Definition

The date the granted copyright commences.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2006-01-02
20050723

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.3.6.2 endDate

Semantic components

Definition

The date the granted copyright expires.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2010-01-02
20120723

Creation/Maintenance notes

Usage notes

Use “OPEN” for an open ended term of restriction. Omit **endDate** if the ending date is unknown or the permission statement applies to many objects with different end dates.

Table of Contents

4.1.4 licenseInformation

Semantic components

Definition

Information about a license or other agreement granting permissions related to an object.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

When **rightsBasis** is “license”, **licenseInformation** should be provided.

Table of Contents

4.1.4.1 licenseDocumentationIdentifier

Semantic components

4.1.4.1.1 licenseDocumentationIdentifierType
4.1.4.1.2 licenseDocumentationIdentifierValue
4.1.4.1.3 licenseDocumentationRole

Definition

A designation used to identify uniquely documentation supporting the specified Rights granted by license within the repository system.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

This semantic unit is intended to refer to a document recording the granting of permission when the Rights basis is license. For some repositories this

may be a formal signed contract with a customer. If the granting agreement is verbal, this could point to a memo by the repository documenting the verbal agreement. The identifier is optional because the agreement may not be stored in a repository with an identifier. For example, in the case of a verbal agreement the entire agreement may be included or described in the **licenseTerms**. If repeated, use **licenseDocumentationRole** to distinguish the role of the given documentation. This replaces the semantic unit in PREMIS version 2.1 which was called **licenseIdentifier**.

Table of Contents

4.1.4.1.1 licenseDocumentationIdentifierType

Semantic components

None

Definition

A designation of the domain within which the license documentation identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Notrepeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/identifiers.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.4.1.2 licenseDocumentationIdentifierValue

Semantic components

None

Definition

The value of the **licenseDocumentationIdentifier**.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

None

Example

http://nrs.harvard.edu/urn-3:HUL.DRS.OBJECT:678

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.4.1.3 licenseDocumentationRole

Semantic components

None

Definition

A value indicating the purpose or expected use of the documentationbeing identified.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

This information distinguishes the purpose of the supporting documentation especially when there are multiple documentation identifiers.

Data Constraint

Value should be taken from a controlled vocabulary.

Example

donor agreement

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.4.2 licenseTerms

Semantic components

None

Definition

Text describing the license or agreement by which permission was granted.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

None

Example

Creation/Maintenance notes

Usage notes

This could contain the actual text of the license or agreement, or a paraphrase or summary of it.

Table of Contents

4.1.4.3 licenseNote

Semantic components

None

Definition

Additional information about the license.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Example

License is embedded in XMP block in file header

Creation/Maintenance notes

Usage notes

Information about the terms of the license should go in **licenseTerms**. **licenseNote** is intended for other types of information related to the license, such as contact persons, action dates, or interpretations. The note may also indicate the location of the license, for example, if it is available online or embedded in the object itself.

Table of Contents

4.1.4.4 licenseApplicableDates

Semantic components

4.1.4.4.1 startDate
4.1.4.4.2 endDate

Definition

The date range during which the license applies or is applied to the content. This is distinct from **termOfGrant**, which applies to a particular act expressed in **rightsGranted** and may differ from the period of time the license, statute or other basis applies to the content.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Specific dates may apply to the particular Rights granted.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

The repository may wish to retain the history of Rights and restrictions associated with the content over time. Associating active dates with particular

Rights bases allows applications to identify which of several rightsStatements are in force at a given time.

Table of Contents

4.1.4.4.1 startDate

Semantic components

None

Definition

The beginning date of the Rights granted.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Example

2006-01-02
20050723

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.4.4.2 endDate

Semantic components

None

Definition

The ending date of the Rights granted.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2010-01-02
20120723

Creation/Maintenance notes

Usage notes

Use “OPEN” for an open ended term of restriction. Omit **endDate** if the ending date is unknown or the permission statement applies to many objects with different end dates.

Table of Contents

4.1.5 statuteInformation

Semantic components

Definition

Information about the statute allowing use of the object.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

When **rightsBasis** is “statute”, **statuteInformation** should be provided.

Table of Contents

4.1.5.1 statuteJurisdiction

Semantic components

None

Definition

The country or other political body enacting the statute.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

The connection between the object and the Rights granted is based on jurisdiction.

Data Constraint

Values should be taken from ISO 3166 if applicable.

Example

us
de

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.5.2 statuteCitation

Semantic components

None

Definition

An identifying designation for the statute.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

None

Example

Legal Deposit (Jersey) Law 200-
National Library of New Zealand (Te Puna Mātauranga o Aotearoa
)
Act 2003 no 19 part 4 s 3

Creation/Maintenance notes

Usage notes

Use standard citation form when applicable.

Table of Contents

4.1.5.3 statuteInformationDeterminationDate

Semantic components

None

Definition

The date that the determination was made that the statute authorized the permission(s) noted.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

The permission in question may be the subject of some interpretation. These assessments are made within a specific context and at a specific time. At another time the context, and therefore the assessment, could change. For this reason it can be important to record the date of the decision.

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2007-12-01
20040223151047.0

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.5.4 statuteNote

Semantic components

None

Definition

Additional information about the statute.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

None

Example

Applicability to web-published content sent for review by counsel 9/19/2008	general

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.5.5 statuteDocumentationIdentifier

Semantic components

4.1.5.5.1 statuteDocumentationIdentifierType
4.1.5.5.2 statuteDocumentationIdentifierValue
4.1.5.5.3 statuteDocumentationRole

Definition

A designation used to uniquely identify documentation supporting the specified Rights granted by statute within the repository system.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

This semantic unit is intended to refer to a document detailing the granting of permission when the Rights basis is statute. If repeated,

use **statuteDocumentationRole** to distinguish the role of the given documentation.

Table of Contents

4.1.5.5.1 statuteDocumentationIdentifierType

Semantic components

None

Definition

A designation of the domain within which the statute documentation identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/identifiers

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.5.5.2 statuteDocumentationIdentifierValue

Semantic components

None

Definition

The value of the **statuteDocumentationIdentifier**.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

None

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.5.5.3 statuteDocumentationRole

Semantic components

None

Definition

A value indicating the purpose or expected use of the documentation being identified.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

This information distinguishes the purpose of the supporting documentation especially when there are multiple documentation identifiers.

Data Constraint

Value should be taken from a controlled vocabulary.

Example

law
application decree
case law

Creation/Maintenance notes

Usage notes

For a particular law one might want to link to various sources of documentation, e.g. the law publication itself (role: law), the application decree that enforces it (role: application decree) or some additional text refining the law by showing a real world verdict (role: case law).

Table of Contents

4.1.5.6 statuteApplicableDates

Semantic components

4.1.5.6.1 startDate
4.1.5.6.2 endDate

Definition

The date range during which the statute applies or is applied to the content. This is distinct from termOfGrant, which applies to a particular act expressed in **rightsGranted** and may differ from the period of time the license, statute or other basis applies to the content.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Specific dates may apply to the particular Rights granted.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

The repository may wish to retain the history of Rights and restrictions associated with the content over time. Associating active dates with particular

Rights bases allows applications to identify which of several **rightsStatements** are in force at a given time.

Table of Contents

4.1.5.6.1 startDate

Semantic components

None

Definition

The date the granted statute commences.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2006-01-02
20050723

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.5.6.2 endDate

Semantic components

None

Definition

The date the granted statue expires.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2010-01-02
20120723

Creation/Maintenance notes

Usage notes

Use “OPEN” for an open ended term of restriction. Omit **endDate** if the ending date is unknown or the permission statement applies to many objects with different end dates.

Table of Contents

4.1.6 otherRightsInformation

Semantic components

Definition

Information about the Rights (other than copyright, license, or statute) that apply to the object(s).

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

This semantic unit is used to supply information about Rights granted when the basis is something other than copyright, license, or statute.

Table of Contents

4.1.6.1 otherRightsDocumentationIdentifier

Semantic components

4.1.6.1.1 otherRightsDocumentationIdentifierType
4.1.6.1.2 otherRightsDocumentationIdentifierValue
4.1.6.1.3 otherRightsDocumentationRole

Definition

A designation used to uniquely identify documentation supporting the specified Rights within the repository system, when the basis for these Rights is something other than copyright, license or statute.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

This semantic unit provides a mechanism to link to documentation for **rightsBasis** values other than those granted through copyright, license, or statute. The Rights basis may be specified in otherRightsBasis.

Data Constraint

Container

Example

Creation/Maintenance notes

The semantic unit is repeatable because there may be more than one document that provides information about the Rights.

Usage notes

This semantic unit is intended to refer to a document recording the granting of permission, the expression of requirements or restrictions, or other information supporting the specified **rightsBasis**. If repeated, use **otherRightsDocumentationRole** to distinguish the role of the given documentation.

Table of Contents

4.1.6.1.1 otherRightsDocumentationIdentifierType

Semantic components

None

Definition

A designation of the domain within which the Rights statement documentation identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/identifiers

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.6.1.2 otherRightsDocumentationIdentifierValue

Semantic components

None

Definition

The value of the **otherRightsDocumentationIdentifier**.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

None

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.6.1.3 otherRightsDocumentationRole

Semantic components

None

Definition

A value indicating the purpose or expected use of the documentation being identified.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

This information distinguishes the purpose of the supporting documentation especially when there are multiple documentation identifiers.

Data Constraint

Value should be taken from a controlled vocabulary.

Example

institutional policy
deed of gift

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.6.2 otherRightsBasis

Semantic components

Definition

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.6.3 otherRightsApplicableDates

Semantic components

4.1.6.3.1 startDate
4.1.6.3.2 endDate

Definition

The date range during which the particular right apply or applied to the content. This is distinct from termOfGrant, which applies to a particular act expressed in rightsGranted and may differ from the period of time the license, statute or other basis applies to the content.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Specific dates may apply to the particular Rights granted.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

The repository may wish to retain the history of Rights and restrictions associated with the content over time. Associating active dates with particular

Rights bases allows applications to identify which of several **rightsStatements** are in force at a given time.

Table of Contents

4.1.6.3.1 startDate

Semantic components

None

Definition

The date the granted right commences.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2006-01-02
20050723

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.6.3.2 endDate

Semantic components

None

Definition

The date the granted right expires.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2006-01-02
20050723

Creation/Maintenance notes

Usage notes

Use “OPEN” for an open ended term of restriction. Omit **endDate** if the ending date is unknown or the permission statement applies to many objects with different end dates.

Table of Contents

4.1.6.4 otherRightsNote

Semantic components

None

Definition

Additional information about the Rights of the object.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/actionsGranted.html

Example

80-year rule

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.7 rightsGranted

Semantic components

Definition

The action(s) that the granting agency has allowed the repository.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.7.1 act

Semantic components

None

Definition

The action the preservation repository is allowed to take.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Value should be taken from a controlled vocabulary. A controlled vocabulary is available at: http://id.loc.gov/vocabulary/preservation/actionsGranted.html

Example

replicate
modify
use
disseminate

Creation/Maintenance notes

Usage notes

It is up to the preservation repository to decide how granular the controlled vocabulary should be. It may be useful to employ the same controlled values that the repository uses for **eventType**.

Table of Contents

4.1.7.2 restriction

Semantic components

None

Definition

A condition or limitation on the act.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

None

Example

No more than three
Allowed only after one year of archival retention has elapsed
Rightsholder must be notified after completion of act

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.7.3 termOfGrant

Semantic components

4.1.7.3.1 startDate
4.1.7.3.2 endDate

Definition

The time period for the permissions granted.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

The permission to preserve may be time bounded.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.7.3.1 startDate

Semantic components

None

Definition

The date the granted permission commences.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2006-01-02
20050723

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.7.3.2 endDate

Semantic components

None

Definition

The date the granted permission expires.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2006-01-02
20050723

Creation/Maintenance notes

Usage notes

Use “OPEN” for an open ended term of grant. Omit **endDate** if the ending date is unknown or the permission statement applies to many objects with different end dates.

Table of Contents

4.1.7.4 termOfRestriction

Semantic components

4.1.7.4.1 startDate
4.1.7.4.2 endDate

Definition

The time period for the restriction granted.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

The current definition of **termOfGrant** is "time period for the permissions granted". This allows repositories to express information about the Rights granted, but some repositories may need to express time-bounded restrictions like embargoes. To express such restrictions, use the semantic units of termOfRestriction: **startDate** for the beginning of the embargo and **endDate** for the end of the embargo.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.7.4.1 startDate

Semantic components

None

Definition

The date the restriction commences.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2006-01-02
20050723

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.7.4.2 endDate

Semantic components

None

Definition

The date the restriction expires.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

To aid machine processing, value should use a structured form. To facilitate exchange of PREMIS-conformant metadata, use of standard conventions, for instance as used in the date elements in the PREMIS schema, is recommended.

Example

2010-01-02
20120723

Creation/Maintenance notes

Usage notes

Use “OPEN” for an open ended term of restriction. Omit **endDate** if the ending date is unknown or the permission statement applies to many objects with different end dates.

Table of Contents

4.1.7.5 rightsGrantedNote

Semantic components

None

Definition

Additional information about the Rights granted.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

A textual description of the granted Rights may be needed for additional explanation.

Data Constraint

None

Example

Creation/Maintenance notes

Usage notes

This semantic unit may include a statement about risk assessment, for example, when a repository is not certain about what permissions have been granted.

Table of Contents

4.1.8 linkingObjectIdentifier

Semantic components

4.1.8.1 linkingObjectIdentifierType
4.1.8.2 linkingObjectIdentifierValue
4.1.8.3 linkingObjectRole

Definition

The identifier of an object associated with the Rights statement.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Rights statements must be associated with the objects to which they pertain, either by linking from the Rights statement to the object(s) or by linking from the object(s) to the Rights statement. This semantic unit provides the mechanism for the link from the Rights statement to an object.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.8.1 linkingObjectIdentifierType

Semantic components

None

Definition

A designation of the domain in which the linking object identifier is unique.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Must be an existing **objectIdentifierType** value.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.8.2 linkingObjectIdentifierValue

Semantic components

None

Definition

The value of the linkingObjectIdentifier.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Must be an existing objectIdentifierValue value.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.8.3 linkingObjectRole

Semantic components

None

Definition

The role of the object associated with an Agent.

Entity information

Obligation	Optional
Repeatability	Not repeatable

Rationale

Data Constraint

None

Example

Creation/Maintenance notes

Usage notes

This value need not be supplied in the ordinary case where the role of the linked-to object is to be governed by the Rights statement. If the object has a different relationship to the Rights statement, however, it should be noted here.

Table of Contents

4.1.9 linkingAgentIdentifier

Semantic components

4.1.9.1 linkingAgentIdentifierType
4.1.9.2 linkingAgentIdentifierValue
4.1.9.3 linkingAgentRole

Definition

Identification of one or more Agents associated with the Rights statement.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Rights statements may be associated with related Agents, either by linking from the Rights statement to the Agent(s) or by linking from the Agents(s) to the Rights statement. This provides the mechanism for the link from the Rights statement to the Agent.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

Linking semantic units are mandatory in the sense that a repository needs to know the information, but are defined as optional because PREMIS does not specify in which direction the linkage should be. In particular, **linkingAgentIdentifier** is optional because a relevant Agent may be unknown, or no Agent may be relevant. The latter is likely when the Rights basis is statute.

Table of Contents

4.1.9.1 linkingAgentIdentifierType

Semantic components

None

Definition

A designation of the domain in which the **linkingAgentidentifier** is unique.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Must be an existing **agentIdentifierType** value.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.9.2 linkingAgentIdentifierValue

Semantic components

None

Definition

The value of the **linkingAgentIdentifier**.

Entity information

Obligation	Mandatory
Repeatability	Not repeatable

Rationale

Data Constraint

Must be an existing **agentIdentifierValue** value.

Example

Creation/Maintenance notes

Usage notes

Table of Contents

4.1.9.3 linkingAgentRole

Semantic components

None

Definition

The role of the Agent in relation to the Rights statement.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

Data Constraint

Values should be taken from a controlled vocabulary.

Example

contact
creator
publisher
rightsholder
grantor

Creation/Maintenance notes

Usage notes

Table of Contents

4.2 rightsExtension

Semantic components

Defined externally

Definition

A container to include semantic units defined outside of PREMIS.

Entity information

Obligation	Optional
Repeatability	Repeatable

Rationale

There may be a need to replace or extend PREMIS defined units.

Data Constraint

Container

Example

Creation/Maintenance notes

Usage notes

For more granularity or to use externally defined semantic units, extensibility is provided. Either local semantic units or metadata using another specified metadata scheme may be included instead of or in addition to PREMIS defined semantic units. When using an extension schema, a reference to that schema must be provided. See further guidance in “Extensibility,” . Either **rightsStatement** or **rightsExtension** must be present if the **Rights** entity is

included. If the **rightsExtension** container needs to be associated explicitly with any PREMIS subunit under Rights, the container **Rights** is repeated. Also, if extensions from different external schemas are needed, Rights should be repeated. It is recommended to give information about the metadata used in **rightsExtension** including date the metadata was created, status of the metadata, internal linking IDs, type of metadata used and its version, message digest and message digest algorithm of the metadata, and type of identifier for external metadata links.

Table of Contents

SPECIAL TOPICS

SPECIAL TOPICS

This section describes several important topics that are too detailed for the Data Dictionary itself. The discussion here provides background information about semantic units and illustrates the thinking behind PREMIS.

Format information

Format information

Formats are an important aspect of digital preservation but before specific semantic units can be defined some fundamental questions need to be addressed including:

- What is a format?
- What types of objects have a format?
- How does one identify a format?
- Is there a difference between a format and a profile?

The concept of format seems almost intuitive, but given the importance of format information to digital preservation PREMIS needs to be very specific about its meaning. The defining feature of a format is the fact that a format has to correspond to some formal or informal specification; it cannot be a random or undocumented layout of bits. Existing definitions do not seem to emphasize this feature sufficiently. Hence, PREMIS uses its own definition: *a specific, pre-established structure for the organization of a digital file or bitstream.*

Format is obviously a property of files, but it can also apply to bitstreams. For example, an image bitstream within a TIFF file may have a format that is defined within the TIFF file format specification. For this reason PREMIS avoids the term “file format” and instead uses the more generic term “ format.”

A preservation repository must record format information as specifically as possible. Ideally, formats would be identified by a direct link to the full format specification. In real implementations an indirect link such as code or string that can in turn be associated with the full format specification is more practical. Format name is a somewhat arbitrary designation that can be used as this indirect link. However, two complications arise when attempting to define the semantic unit(s) to be used as this link.

First, format designations in common use, such as MIME types and filetype extensions, are not granular enough to be used in this way without the addition of version information. One possible way to deal with this is to include the version in the format name (e.g., “TIFF 6.0”). The alternative is to define two

semantic units, one for name and one for version. To allow existing authority lists such as MIME type to be used it was decided to use the latter approach. Hence, in the Data Dictionary *formatDesignation* has two components: *formatName* and *formatVersion*.

Second, centrally maintained format registries are expected to be the best way to get detailed format information in the future.[32] In the PREMIS model the format name provides an indirect link to the format specification. In the registry environment not one but two things must be known: what registry is being used, and what identifies the specification within the registry. One way to deal with this is to combine all format identification into a single set of semantic units. The alternative is to define different containers for registry and non-registry environments. A good argument for a single set is that a repository that uses its own authority list of format names to associate digital objects with specifications is, in essence, maintaining its own format registry, where the identification of the registry itself is simply assumed. However, with major format registries still under development it was decided that it was dangerous to make assumptions about what would be needed to use them. Ultimately, two containers were defined: formatDesignation and formatRegistry.

Within one *format* container it is mandatory that at least one of these two semantic units be present to provide the necessary identifying information. They are more explicitly linked when used together.

It was also decided to make *format* repeatable to allow for the cases where (a) more than one registry is in use, or (b) resolving format identification is not immediately possible, or (c) more than one equally specific format designation applies.

(a) If multiple registries are used, repeatability of the *format* element makes it possible to clearly record inconsistencies between the formats identified by each registry. To reduce ambiguity, *formatRegistryRole* should be used to indicate for which particular purpose a registry is being used: e.g. format identification, format validation, characterization, profile identification. Exactly one registry should be indicated by the *formatRegistryRole* as the authoritative source for identifying formats. *formatNote* should be used to record supplementary, qualifying information, e.g. when several identifications are true in conjunction [e.g. BWF and WAV].

(b) In practice, running tools for file identification may produce several candidate identities per file or bitstream and resolving format identification may not be immediately possible.[33] Repeatability of the *format* element makes it possible to capture them. *formatNote* should be used to record

32 See, for example, PRONOM at http://www.nationalarchives.gov.uk/pronom,the Unified Digital Format Registry at http://www.udfr.org/.

33 Such tools include DROID (http://www.nationalarchives.gov.uk/information-management/manage-information/preserving-digital-records/droid/), TRiD (http://mark0.net/soft-trid-e.html), and Apache Tika (http://tika.apache.org/).

supplementary, qualifying information, when several identifications form a disjunction of candidate formats [e.g., TIFF 3.0 or TIFF4.0]. It is not uncommon for particular implementations of formats, often called profiles, to be specified. For example, GeoTIFF (for geographic images), TIFF/EP (for digital cameras), and TIFF/IT (for prepress images) are compatible with the TIFF specification, but narrow it by requiring certain options, or extend it by adding tags. Because of this it is possible for a file to have more than one format, for example, both TIFF and GeoTIFF. There are various possible options to accommodate this, such as recommending that both be recorded, or defining a separate semantic unit for format profiles. However, in such cases, PREMIS recommends recording the most specific format designation that applies. Current format registries (e.g. PRONOM and UDFR) record format profiles, extensions, and modifications as separate formats and indicate the relationships among them. It is recognized that the most specific designation is a matter of opinion and will be implementation specific. For example, for a METS document (that is, an XML instance conforming to the METS schema) one repository may consider XML to be the most specific format, while another may consider METS to be the most specific format.

(c) In some cases, a file or bitstream will be found to conform to more than one format specification, where each is equally specific (that is, neither is a proper subset of theother). In this case, each of the formats should be recorded separately. Multiple formats may also be recorded if it is important to indicate the version of each.

Environment

Environment

Digital materials are distinctly different from analog materials because a complex technical environment is interposed between user and content. Application software, operating systems, computing resources, and even network connectivity allow the user to render and interact with the content. Separating digital content from its environmental context can make the content unusable. Therefore, careful documentation of the technical environment associated with an archived digital object can be an essential component of preservation metadata.

With the advent of Intellectual Entities in PREMIS 3.0, the way in which environments are modeled in PREMIS has been transformed. Before version 3.0, there was an *environment* container within an Object that described the environment supporting that Object. If a non-environment Object needs to refer to an environment, it is now recommended that the environment is described as an Object in its own right and the two Objects are linked with a dependency relationship. The environment Object could be conceptual (i.e. an Intellectual

Entity) or could refer to a digitally stored representation, file or bitstream. Unlike digital representations, physical representations cannot be broken down into files. A physical object can be described using an Intellectual Entity, while the physical object itself is a representation of this Intellectual Entity. An environment representation may describe either physical, tangible items, such as a concrete physical instance of a floppy drive, or intangible items, such as the software driver for it. A key result of this is that each environment can be described and preserved independently.

We may take as an example the need to record that a non-environment Object (a content Object) should use a particular piece of hardware, an operating system and some application software to be rendered. Such information can be recorded by linking the non-environment Object to three different environment Intellectual Entities which would separately describe the hardware, operating system and software application. In the latter two cases, it is possible that these could be held as (non-environment) representations, files or bitstreams in the repository (e.g., the operating system could be a disk image and the software application could be an executable of some sort). The non-environment Objects might link directly to environment Intellectual Entities Objects as outlined in Figure 5, or the link could be indirect (i.e. it links to an environment Intellectual Entity which in turn links to the digital instantiation of that environment), as outlined in Figure 6.

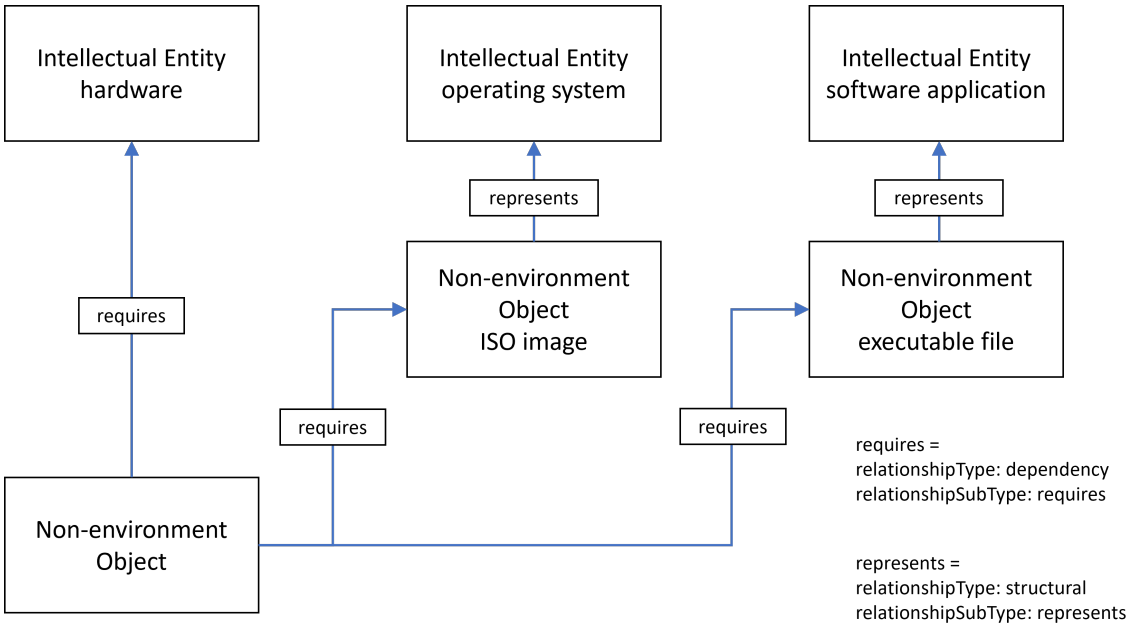

Figure 5. An object and its rendering environment: direct link

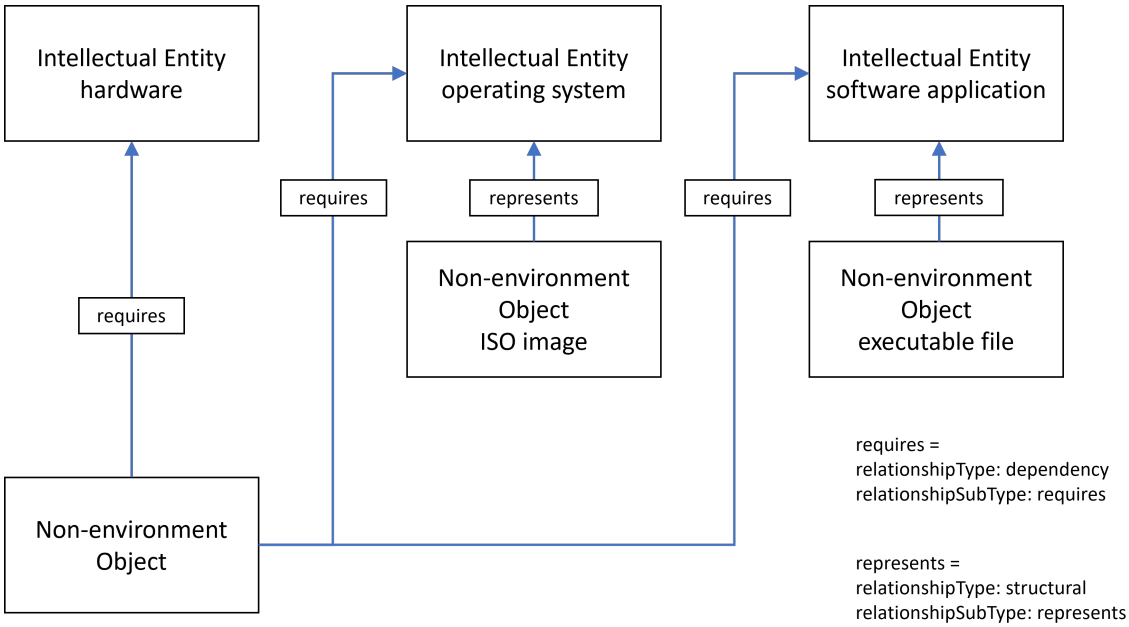

Figure 6. An object and its rendering environment: indirect link

This way, each part of the environment stack can be independently described and related to its own dependencies, as outlined in Figure 7.

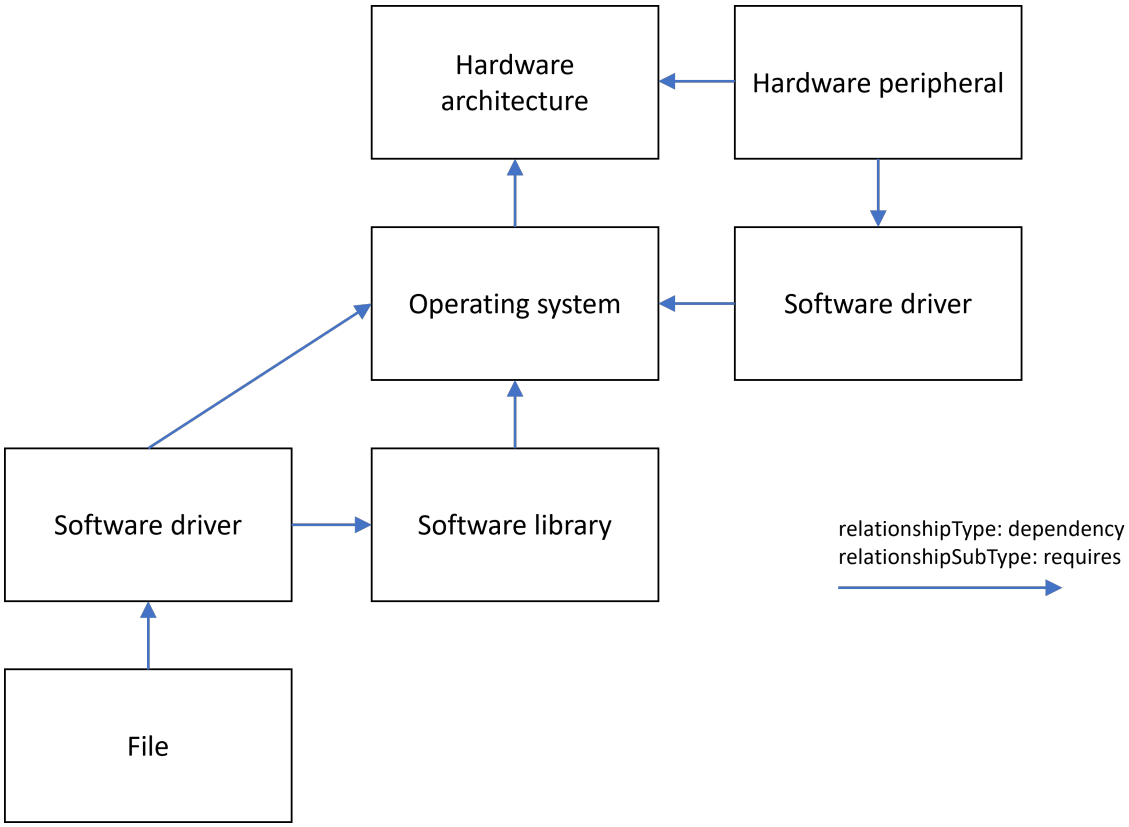

Figure 7. Parts of an environment stack and dependency relationships between them

One should also record the **purpose** of an environment in a particular context (*create*, *render*, *edit*) and an assessment of the extent to which the described environment supports its purpose(*minimal*, *recommended*, *known to work*), which PREMIS calls its *characteristic*. This information is context dependent, because the same environment can be used to create or render objects in different contexts. Therefore this information is not an attribute of an environment, but rather an association between an environment and an Object that it supports in some way. To record such information accurately, the *relationship* semantic container has been enriched with two additional semantic components: *relatedEnvironmentPurpose* and *relatedEnvironmentCharacteristic*.

To enable this to work it has been necessary to introduce two new semantic units to Object that only apply to Intellectual Entities that model environments: *environmentFunction* and *environmentDesignation*.

The first of these (*environmentFunction*) allows for a description of the type of the environment. This classification can be at any required level of specificity. At the most generic level all environments are either hardware or software; when one refines these types, you can have more specific values (e.g. chip set or peripheral might be 2nd level functions where the 1st level was hardware, while operating system or application software might be 2nd level functions where the 1st level was software). To allow implementers to record as many levels of specificity as they wish, the *environmentFunction* semantic container is repeatable, with the actual category being recorded in *environmentFunctionType* and their level of specificity recorded in *environmentFunctionLevel*. The implementer can choose only to record the most specific type, in which case the level is optional (as only one is recorded).

The second of these (*environmentDesignation*) is designed to allow a clear description of the environment. However, when making reference to an environment it is sometimes desirable to be very generic (e.g. a range of versions, like “Firefox version 2 or later”) and sometimes to be very specific (e.g., a particular release of a specialist piece of application software with a given patch set applied). Hence, the *environmentDesignation* semantic unit must accommodate both needs. To achieve these, the semantic unit is split into three main subsidiary parts: *environmentName*, *environmentVersion*, *environmentDesignationNote* as well as allowing for any extension via the *environmentDesignationExtension* semantic unit.

Of the three main subsidiary parts of *environmentDesignation*, only *environmentName* is mandatory and the others are optional. A very generic environment would only have this first element set: for example, to model any version of an internet browser, an Intellectual Entity could be created with the *environmentName* set to be simply the name of the internet browser and no *environmentVersion* nor *environmentDesignationNote* nor *environmentDesignationExtension*.

Designating a specific environment: implementation considerations

Designating a specific environment: implementation considerations

The distinction between name and version can be vague in some cases. Sometimes, a new version of a software or hardware product becomes a standalone product in its own right. For instance, Windows XP is, strictly speaking, a version of the Windows NT operating system architecture. But in its software roadmap, it is rather considered an independent product with itsparticular version numbers and lifecycle. If one wants to describe Windows XP Professional Service Pack 2, the following implementation options are possible:

1. Record Windows NT in *environmentName* and Windows XP Professional Service Pack 2 in *environmentVersion*. Here, Windows NT is considered the primary product being described.
2. Record Windows XP in *environmentName*, and record Windows XP Professional Service Pack 2 in *environmentVersion*. Here, the primary product is considered to be Windows XP.
3. Record Windows XP Professional in *environmentName* and Service Pack 2 in *environmentVersion*. Here the primary product is considered to be Windows XP Professional.

The three implementation options are valid, though 2 or 3 are considered the most relevant from a preservation standpoint, as the level of granularity of the product must capture the characteristics used in preservation watch. For example, the preservation of the operating system relies very much on the lifecycle roadmap of the product within Microsoft. This is, in this example, clearly relevant at the Windows XP level (2) or, even more so, at the level of a particular edition (3). Thus, implementation option (3) should be the most satisfactory way to record that, all other things being equal.

Another relevant example is the *JHOVE* file analysis software: even though *JHOVE 2* declares itself as the second major version of the *JHOVE* software application, it is a completely different product with a different architecture, different features, a different maintaining agency, and an independent lifecycle (*JHOVE 1* and *JHOVE 2* continue to evolve in parallel). For these reasons, it would be better to have *JHOVE 2* in *environmentName*, and record the specific build number of the software preserved or used in the repository in *environmentVersion*.

In any case, an implementer should define their own naming policy to express name and version information and apply it consistently for all their environment descriptions.

Handling different levels of specificity at the same time: a possible strategy

Handling different levels of specificity at the same time: a possible strategy

For example, suppose there is a need to model the Ubuntu operating system and we need to reference version 12.0 and the 32-bit version. If there is a need to reference both generic and specific versions of this operating system, it is likely that a decision would be made to create a hierarchy of Intellectual Entities and maintain each *environmentDesignation* element throughout the hierarchy, simply adding more elements as specificity increases. So, a top-level Intellectual Entity could be created where only *environmentName* has been set (to “Ubuntu”), a 2nd level Intellectual Entity could be created where, in addition, the *environmentVersion* has been set (to “12.0”) and a 3rd level Intellectual Entity could be created where, in addition, the *environmentDesignationNote* has been set (to “32 bit”). The links between these entities would be modeled using the *relationship* semantic unit to link each level to its parent level. This would provide a natural hierarchy where other versions of the operating system can easily be added (e.g., by adding another 3rd level Intellectual Entity for a “64 bit” version which would be linked to the same 2nd level Intellectual Entity as its “32 bit” sibling). In this particular arrangement, it is possible to reference any Ubuntu operating system, any version of the Ubuntu operating system and specific operation systems. If it was decided that there was a need to refer to 32-bit Ubuntu operating systems across versions a slightly different hierarchy could be created. Equally, further refinement would be needed (e.g., expanding *environmentDesignationNote*) if it was felt that more specificity was needed (e.g., recording patch releases etc.).

In addition, as digital environments are made up of components that can be broken down into smaller and smaller components, their descriptions can easily become extremely complex. Simultaneously, itis likely that these descriptions will tend to be the same for entire classes of digital objects, for example, for all Files of a particular format. The combination of these two factors suggests that the most efficient model for collecting and maintaining environment metadata is a centralized registry (or series of registries). Some registries do exist but they are not necessarily complete (e.g., they are limited to a certain kind of environment and/or to a specific domain,

like web archiving) and they also have a mixture of generic and specific definitions of entities.[34] Hence, PREMIS allows the modeling of environments both with reference to an external registry and without. To allow a registry to be referenced, the *environmentRegistry* semantic unit should be used. This allows the registry to be specified (via *environmentRegistryName*), the identifier for the corresponding environment in that registry to be specified (via *environmentRegistryKey*) and the role of the registry to be clarified (via *environmentRegistryRole*). As these semantic units are conceptually identical to the way the *formatRegistry* works, they will not be explained here in much detail. However, it is worth mentioning that the *environmentRegistryRole* could be used to document the potential mismatch between a generic registry entry and a more specific Object created in PREMIS (i.e. to record that the registry role is describing a more generic entity that the PREMIS Object is modeling).

In most cases, environments will not be held to render individual bitstreams, since software operates on known file formats, or in the case of compound objects, on aggregations ofknown file formats. However, it could apply to bitstreams in some situations. For instance, it is possible for a single AVI file to be used as the common container for video streams each requiring the use of specialized rendering software. In an AVI file encapsulating heterogeneous bitstreams, each of the bitstreams may require a substantially unique preservation workflow. Setting the environment at the bitstream level maintains the important association that a particular bitstream requires a particular environment. If the environment were set at the file level, this association would be lost, complicating preservation efforts that require the disaggregation of the file.

In other cases a file format may contain two or more discrete bitstreams with wholly different semantics, but software designed to support the format may be able to correctly interpret and/or render any bitstream appearing within the file. For example, a TIFF viewer rendering an image knows to skip past the header information (a bitstream within the file) to reach the image data (a second bitstream within the file). It is not always necessary to link to separate environment information for each of these bitstreams if they are both handled by any rendering application compatible with the TIFF format specification.

It is also worth noting that a representation may relate to a different environment than the individual files that constitute that representation. For example, a browser is appropriate for rendering a multimedia web page consisting of text, static images, animation, and sound components, but each individual file rendered separately would require different environments than the one for the compound object (representation) as a whole.

34 See for example the TOTEM Registry delivered as part of the KEEP project, http://www.keep-totem.co.uk/. PRONOM also contains an amount of software information, http://apps.nationalarchives.gov.uk/PRONOM/Default.aspx.

Note that PREMIS is not attempting to hold all the information needed to determine the mechanism(s) by which archived objects are delivered from the repository to the user (e.g., over a network, on CD, on DVD, etc.). This is not seen as a core preservation process. Moreover, the usefulness of a delivery mechanism description will likely vary from repository to repository, depending on local dissemination policies. However, the environment information held in PREMIS could help such dissemination mechanism make a decision about how to deliver content.

Object characteristics and composition level: the “onion” model

Object characteristics and composition level: the “onion” model

When an object is compressed or encrypted, the format of the object is determined by the compression or encryption scheme. At the same time, the object has an underlying format that is different. Objects such as these pose the problem of how to describe complex layers of encodings and encryptions so that they can be reversed correctly. PREMIS uses the metaphor of an onion: a digital object can be wrapped in layers of encodings that need to be “peeled off” in a particular sequence. The onion model is implemented by treating each layer as a “composition level,” and organizing metadata into sets of values pertaining to each layer.

The simplest example is a single file with no additional encoding or encryption (beyond that inherent in its format). In this case there would be one instance of the semantic unit *objectCharacteristics* with *compositionLevel* value of 0 (zero). The object characteristics of a simple PDF, for example, might include a message digest, a size of 500,000 bytes, a format of PDF 1.2, inhibitors such as no printing allowed, and a creating application of Adobe Acrobat. If a compressed version of that PDF file were created using the UNIX gzip utility and stored in the repository, the compressed file would be described with two *objectCharacteristics* blocks. The first, with *compositionLevel* zero, would be the same as for the simple PDF, and the second with *compositionLevel* 1, would record another message digest, a smaller size, and a format of gzip. This could continue for as many layers as necessary to describe the object completely.

To extract the content object, one works backwards through the composition levels from highest to lowest, using an application appropriate to the format of the layer. In the example above, to get to the PDF one applies a tool that understands the gzip format. Having un-gzipped the content, it can be

compared to the size and fixity information previously stored to determine that the correct object has been extracted. (In practice, some of the encodings have checking mechanisms built in.)

Note that this model assumes that the object is being stored with the composition layers preserved. If the archive has already removed the layers and is storing the base object, the information about the removal of the layers is Event data rather than composition data. That is, if a decompressed version of object A is created and called object B, A is related to B by a derivation relationship (sourceOf) with a related decompression Event.

To extract the content object, one works backwards through the composition levels from highest to lowest, using an application appropriate to the format of the layer. In the example above, to get to the PDF one applies a tool that understands the gzip format. Having un-gzipped the content, it can be compared to the size and fixity information previously stored to determine that the correct object has been extracted. (In practice, some of the encodings have checking mechanisms built in.)

Note that this model assumes that the object is being stored with the composition layers preserved. If the archive has already removed the layers and is storing the base object, the information about the removal of the layers is Event data rather than composition data. That is, if a decompressed version of object A is created and called object B, A is related to B by a derivation relationship (sourceOf) with a related decompression Event.

Bitstreams and filestreams are not composition layers. If an archive chooses to manage bitstream or filestream objects, they are separate objects whose storage location is at an offset inside a file, which is itself a separate object with characteristics and metadata and its own storage location. Each of these may have composition layers including encryption and encodings. The level-zero composition layer of the file would be the file without additional encryption or encoding (beyond that inherent in its format); that a bitstream inside that file is a managed object is a separate issue (and object) distinct from the layers of encodings of the file.

Formats such as tar and ZIP that can bring together (“package”) several files into one file present a related but not identical problem. If the package consists of only one object, one could treat the package as yet another composition layer; for example, a file that is encrypted, then zipped would have three composition levels. If the package contains more than one file, however, it should be treated as a separate object that provides the storage location for the contained objects so that there can be a distinct metadata record for each of the contained objects. For example, a ZIP file containing two PDF files should be treated as three objects: the ZIP file with a base composition format of ZIP, and two other objects whose storage location is inside the ZIP file. As with bitstreams, the objects inside the ZIP file object are logically distinct from the

containing object. They each may have completely different sets of metadata and indeed may have additional composition layers as well. One could imagine an encrypted ZIP file containing two files that are themselves each separately encrypted. There would then be three objects, each with two composition levels.

With PREMIS 3, it is possible to record the creating application of an Object as a discrete environment Object with a purpose of “create”. However, this new mechanism does not tie the environment to a specific composition level, but to the Object as a whole: in the example above of a gzip-compressed PDF file, this would not be possible to specify that the creation environment “UNIX gzip utility” is tied to the *compositionLevel* number “1”, and that the creation environment “Adobe Acrobat” is tied to the *compositionLevel* number “0”. For those reasons, it is recommended to use *creatingApplication* whenever such expressivity is required.

In some cases the composition level cannot be isolated, e.g. it is not possible to automatically detect if a File is encrypted or not. In such cases, it is recommended to explicitly state that the *compositionLevel* is unknown as it is a tangible risk for the preservation of the Object.

Fixity, integrity, authenticity

Fixity, integrity, authenticity

A digital preservation metadata standard needs to give considerable attention to the concepts of fixity, integrity, and authenticity of digital objects. Objects that lack these features are of little value to repositories that have the mission to protect evidentiary value or indeed to preserve the cultural memory.

In the PREMIS Data Dictionary the information needed to verify fixity (that an object is unchanged since some earlier point in time) is described by a set of semantic components under the semantic unit *objectCharacteristics*. Running a fixity check program on an object to detect unauthorized changes to it is detailed as an event. In the analog world, acts of publication and production serve to fix an object in time. In the digital domain, hash algorithms that create a message digest can be used to implement a fixity check for an object. If the message digest created by an algorithm at one point is identical to the message digest created by the same algorithm at a later point, this indicates the object did not change during the interim. In fact, it is common to create and test two or more message digests using different algorithms to be certain that an object is fixed.

While this procedure can indicate with some confidence that an object has not changed over time, it does not address the object’s integrity or authenticity.

In the PREMIS model, verifying the integrity of an object is considered an event. Format identification and validation are key indicators of the integrity of a file. Software technology such as JHOVE can verify that a format is what its file extension claims, as well as determining the level of compliance to a particular format specification.[35] The integrity of a representation may have to be verified by special programs that understand the structure of the representation. If the representation includes structural metadata, the structural metadata can be used to test that all files are present and named correctly.

In some cases it is necessary to be pragmatic about enforcing integrity. For example, a lot of filesthat purport to be HTML are not strictly compliant with the format specification (e.g., failing to close tags). From a purist point of view these files could be rejected from a repository. On one hand, this would mean excluding a lot of files which most browsers could render, since browsers are tolerant of such failings in current practice. On the other hand, there is no guarantee that future browsers will be equally tolerant especially as more modern ways of producing HTML tend to produce files that are strictly compliant with the formal specification. In any case, it is essential to record the fact that the file is not compliant to the format specification. PREMIS allows such imperfections in determining integrity to be recorded either in a *formatNote* semantic unit or as the outcome of a validation event (see “Quirks and anomalies” in “Non-core metadata”).

The authenticity of a digital object is the quality of being what it purports to be. As the Digital Preservation Coalition (DPC) explains, “In the case of electronic records, [authenticity] refers to the trustworthiness of the electronic record as a record. Confidence in the authenticity of digital materials over time is particularly crucial owing to the ease with which alterations can be made.”[36]

Authentication, or the demonstration of authenticity, is multifaceted, and is, ultimately, a matter of human judgment often relying on the skills of an archivist, librarian or other trained individual. However, they need evidence to help them in this work including both technical and procedural evidence. Such evidence may include the maintenance of detailed documentation of digital provenance (the history of the object), the preservation of a copy of the object that bit-wise is identical to the content as submitted, the preservation of a representation that retains all the characteristics identified as important (significant properties), and the use of digital signatures. PREMIS metadata supports the documentation of provenance by defining semantic units associated with Events and allowing linking between Object entities and Event entities. Fixity can be tested against stored message digest information and the testing itself recorded as an event. Digital signatures can also aid authentication and are discussed next.

35 JHOVE – JSTOR/Harvard Object Validation Environment, http://jhove.openpreservation.org/.
36 Digital Preservation Handbook, http://www.dpconline.org/advice/preservationhandbook.

Digital signatures

Digital signatures

Digital signatures rely on an external (mutually trusted) authority to verify the signature. This is relatively new technology with a variety of authorities and exact methodologies and there is noguarantee that the verification service used will remain available in the very long term. Hence, as is described in more detail below, they provide a potential problem for repositories aiming to provide long-term digital preservation.

Preservation repositories use digital signatures in three main ways:

- For **submission** to the repository, an Agent (author or submitter) might sign an object to assert that it truly is the author or submitter.
- For **dissemination** from the repository, the repository may sign an object to assert that it truly is the source of the dissemination.
- For **archival storage**, a repository may want to archive signed objects so that it will be possible to confirm the origin and integrity of the data.

The first and second usages are common today as digital signatures are used in the transmission of business documents and other data. Typically, validation takes place shortly after signing and there is no need to preserve the signature itself over time. In the first case the repository may record the act of validation as an event, and save related information needed to demonstrate provenance in the event detail. In the second case the repository might also record the signing as an event but the use of the signature is the responsibility of the receiver. Only in the third case, where digital signatures are used by the repository as a tool to confirm the authenticity of its stored digital objects over time, must the signature itself and the information needed to validate the signature be preserved.

Just as with a pen-and-ink signature or seal, reliable digital signatures require that:

- The process of producing a signature is considered to be unique to the producer.
- The signature is related to the content of the document that was signed.
- The signature can be recognized by others to be the signature of the person or entity that produced it.

To create a digital signature, first a secure hash algorithm (SHA) is applied to content (a file or bitstream) and used to produce a short message digest from that content. The message digest and, optionally, related information are then encrypted using asymmetric cryptography. Asymmetric cryptography is based on using a pair of keys: a private key to encrypt and a public key to

decrypt. The private key must be held secretly and securely by the signer, ideally in secure hardware. This accomplishes the goal of a signature unique to the producer. Since the message digest that is encrypted is tied directly to the content this also accomplishes the goal of relating the signature to the content. The signature can be verified by decrypting the signature with the signer's public key and comparing the now-decrypted digest with a new digest produced by the same algorithm from the same content. If the content had been changed, the comparison would fail.

The goal of connecting the signature to the signer is based on establishing trust between Agentsthat may be internal or external to the repository. For example, agent A ought to trust a signature by agent B if a third party trusted by A asserts that the signature is truly B's. This principle governs notarization of written signatures. The same approach is used in digital signatures, where a trusted third party certifies that a particular key is indeed the public key of the signer. This extends to a chain of trust, whereby the trusted body trusts an intermediary which in turn certifies the signer's public key. This process is typically, but not necessarily, implemented using X.509 certificates, or certificate chains.

This is important for preservation, because the standard current mechanisms for establishing trust in a certificate relies on a set of services that are not likely to be available for the long term. For preservation, widely sharing and safely storing the public key as a formal document may be a more suitable approach. For example, a university might regularly publish its public key in its annual report and make it available on its Web site.

Digital signature metadata

Digital signature metadata

For a preservation repository to validate a digital signature at a later date, the repository will need to store:

- The digital signature itself.
- The name of the hash algorithm and encryption algorithm used to produce the digital signature.
- The parameters associated with these algorithms.
- The chain of certificates needed to validate the signature (if a certificate model is used to relate the signer and the signer's public key).

It is recommended that a repository also store the definitions of the algorithms and relevant standards (e.g., for encoding the keys) so that these methods could be re-implemented if necessary.

The W3C's *XML-Signature Syntax and Processing (XMLDsig)* is a de facto standard for encoding digital signatures that provides a clear functional model for them.[37] PREMIS adopted the names and structure of semantic units from that specification where applicable. However, *XMLDsig* is both too generalized and too specific to be universally applicable in a preservation context. It is too generalized because it allows multiple data objects (files and/or bitstreams in the PREMISmodel)to be signed together, while in the PREMIS model a digital signature is a property of a single object. It is too specific because it prescribes a particular encoding and validation methodology that is not universally applicable.

The Data Dictionary defines the following structure:

1.8 signatureInformation (O, R) [File, Bitstream]
  1.8.1 signature (O, R)
    1.8.1.1 signatureEncoding (M, NR) [File, Bitstream]
    1.8.1.2 signer (O, NR) [File, Bitstream]
    1.8.1.3 signatureMethod (M, NR) [File, Bitstream]
    1.8.1.4 signatureValue (M, NR) [File, Bitstream]
    1.8.1.5 signatureValidationRules (M, NR) [File, Bitstream]
    1.8.1.6 signatureProperties (O, R) [File, Bitstream]
    1.8.1.7 keyInformation (O, NR) [File, Bitstream]
  1.8.2 signatureInformationExtension (O, R) [File, Bitstream]

The hash and encryption algorithms employed are recorded in *signatureMethod*; for example, "DSA-SHA1" would indicate the encryption algorithm is DSA and the hash algorithm is SHA1. The digital signature itself is the *signatureValue*. Information about the generation of the signature that is not needed to validate the signature (e.g., the date and time the signature was generated) is stored in *signatureProperties*. The public key used to validate the signature is indicated in *keyInformation*. Since there are many types of keys each with different structures, these structures were not defined in the Data Dictionary and implementers will need to use externally defined structures. For this reason, *keyInformation* is defined as an extensible container. Where they apply, repositories are encouraged to use the *XMLDsig* "KeyInfo" definitions in *keyInformation*.

The semantic units discussed above have analogs in the *XMLDsig*:

PREMIS - XMLDsig

signatureMethod - <SignedInfo> <SignatureMethod>
signatureValue - <SignatureValue>
signatureProperties - <Object> <SignatureProperties>
keyInformation - <KeyInfo>

37 XML-Signature Syntax and Processing, W3C Recommendation 12 February 2002, http://www.w3.org/TR/xmldsig-core/.

Three semantic units not included in XMLDsig were added to the Data Dictionary: *signatureEncoding*, *signer*, and *signatureValidationRules*. The semantic unitsignatureEncoding indicates the encoding of the values of the subsequent semantic units; this is not included in *XMLDsig* because that document mandates a particular encoding, which cannot be assumed in a broader context. The name of the signer can be extracted from the signer's certificate if this is included in *keyInformation*, but isolating this information in *signer* makes it easier to access. Documentation of the process to be used in validating the signature is stored or pointed to in *signatureValidationRules*. As with *signatureEncoding*, this is not in *XMLDsig* because *XMLDsig* requires a particular validation method.

In cases where a repository is able to use *XMLDsig* and prefers to do so, the entire schema can be used in place of the PREMIS *signature* container, via the extension container *signatureInformationExtension*. In this case the mandatory PREMIS elements are either mandatory in *XMLDsig* (*signatureMethod*, *signatureValue*) or implied by the requirements of the *XMLDsig* specification (*signatureEncoding*, *signatureValidationRules*). In cases where a repository cannot use or chooses not to use *XMLDsig*, it can still use the "KeyInfo" elements defined in the *XMLDsig* schema to define the semantic units recorded in *keyInformation*.

Non-core metadata

Non-core metadata

PREMIS does not attempt to include every possible metadata concept in the Data Dictionary. Unless otherwise noted this does not imply that these semantic units are not necessary or important in other contexts. For specific implementations there may be legitimate reasons to record this information in some form.

Aggregation: Aggregation means the embedding of objects into a larger object (rather than a collection of discrete objects). The property of being an aggregate can be inferred from the presence of multiple files and/or bitstreams, which can be inferred by the presence of whole/part structural relationship between objects belonging to the same category. That semantic unit makes no distinction between an aggregation that is ingested and an aggregation that is created by the preservation repository for storage or other purposes; however, this distinction was not felt to be core.

Quirks and anomalies: *A Metadata Framework to Support the Preservation of Digital Objects* [1] defines "quirks" as "any loss in functionality or change in

1 A Metadata Framework to Support the Preservation of Digital Objects (Dublin, Ohio: OCLC Online Computer Library Center, 2002), http://www.oclc.org/research/projects/pmwg/pm_framework.pdf.

the look and feel of the Content Data Object resulting from the preservation processes and procedures implemented by the archive." "Anomalies" describe aspects of an object that do not meet the specification for the object. Quirks and anomalies could be held as either the outcomes of Events or classified as properties of Objects.

Object resulting from the preservation processes and procedures implemented by the archive." "Anomalies" describe aspects of an object that do not meet the specification for the object. Quirks and anomalies could be held as either the outcomes of Events or classified as properties of Objects.

The argument for treating these as outcomes of events is that quirks by definition result from an event, and anomalies are discovered through the event of validation. If treated this way, an anomaly would be recorded as part of the description of a validation event; the semantic unit *eventOutcome* would indicate problems, and the semantic unit *eventOutcomeDetail* would record the known anomalies.

An argument for treating quirks and anomalies as properties of an object is that this appears to elevate them in importance and gives them a direct as opposed to indirect association with the object.

The decision is arbitrary. The Data Dictionary generally treats quirks and anomalies as outcomes of events, recorded in *eventOutcomeDetail*. However, some specific "anomalies" can also be recorded in other PREMIS entities if this is more appropriate (e.g., deviations from format specifications can be recorded in the *formatNote* semantic unit as discussed above).

Byte order: Byte order determines whether numbers of more than eight bits are stored from most to least significant ("big-endian") or from least to most significant ("little-endian"). Byte order is hardware dependent and can cause problems when data is shared between different types of computers. However, it does not pertain to all formats. For example, it is irrelevant for encodings such as ASCII, where one byte equals one character, and UTF-8, which is byte-order independent. Thus, byte order might better be treated as format-specific technical metadata. It is noted that ANSI/NISO Z39.87 (Data Dictionary – Technical Metadata for Digital Still Images) [38] includes byte order as technical metadata for images.

Character encoding: This item of metadata is important, but it is format-specific technical metadata, useful only for text files and files that can include text.

Dissemination format: PREMIS is based on the premise that the "preservation format" is the object of preservation activity, which may or may not be the same as the dissemination format. Whether or not the preservation

38 ANSI/NISO Z39.87 - Data Dictionary - Technical Metadata for Digital Still Images 2006 (R2011).

format is immediately renderable or is transformed for dissemination is an implementation choice. For example, if the preservation format is a TIFF image, one preservation repository might create a dissemination version (say a JPEG image) on the fly for user access, while another repository might deliver the TIFF master. A third repository might store and process both the TIFF master and the JPEG access copy.

The Data Dictionary does not address the creation of metadata objects that are not stored in a preservation repository. This is based on the assumption that while a dissemination format is important to a repository operationally, it is not core to preservation processes.

Embedded metadata: Some formats allow file objects to contain embedded metadata. The Data Dictionary does not contain any means of indicating this for now, with the understanding that this will probably have to be revisited in the next several years as more and more formats include embedded metadata. For the time being, if embedded metadata is extracted and stored elsewhere, there is no need to note the existence of embedded metadata in the file.

It is also recognized that there is a distinction between standard embedded metadata defined by a file format and locally defined metadata that might be inserted into a file header. Any local divergences from standard formats will likely need to be documented as anomalies.

Event type: The semantic unit *eventType* is core, but not all types of events were considered core, and some were deliberately omitted from the list of suggested values provided in the Data Dictionary. In particular, microfilming (preservation reformatting), moving a file offline, and media refreshment are not core events. Events likely to be handled by a storage system, such as mirroring or the creation of backup copies, would probably be recorded in a system log and are not raised to the level of an event that has metadata associated with it.

Event next occurrence: Many actions taken by a preservation repository are performed periodically, for example, daily or weekly monitoring actions. It could be useful to record an action date or “tickler” for the next scheduled occurrence of an event. This is considered a matter of repository policy and implementation, and not a core property of events.

File pathname/URI: This element was seen as both implementation specific and system dependent. It was not seen as information that would be explicitly recorded in a repository. Often the pathname or location of an object is not known in a content management system; only the unique object identifier of the asset is known and needed for retrieval. Alternatively, in some systems, such as the Handle system, the *objectIdentifier* alone is sufficient for retrieving the file. Therefore, a broader, less system-dependent semantic unit was defined: *contentLocation*. It can be interpreted narrowly (a value could be an exact path

or a “fully qualified” path or filename) or broadly (any information needed to retrieve a File from a storage system, which may include information used by a resolution system such as the Handle system).

Global identifier: PREMIS does not consider the distinction between an externally known identifier and an internally known identifier to be significant. An internal identifier could easily become known outside of the repository and then would be a global identifier. This raises the issue of whether internal identifiers would be sufficiently unique in an external context to function as a global identifier. However, as the **objectIdentifier** always includes an identifier type as well as value, the combination of type and value would be unique even if the type were some local repository scheme.

MIME type: The Internet Media Type and SubType (commonly called “MIME type”) is subsumed under *formatDesignation*. Format designation is intended to be more granular and precise than MIME type and includes multiple format identification schemes, of which MIME type can be one. A MIME type alone is not rigorous enough to identify formats for digital preservation: not all formats have MIME types, it is too coarse a typing mechanism, it is not necessarily current, and it provides no versioning information. Good practice is to include format name and version and use MIME type only if no other data is available.

Modification date: The PREMIS data model asserts that metadata describes only one Representation, File or Bitstream object at any given time. If such an object is changed or modified, a new object is created that is related to the previous one. Each new object then has its own set of metadata, and the relationship between the two is also described. The model does not allow for modifying a representation, file or bitstream object and keeping a set of metadata that describes a history of changes about that object. Therefore, there would be no modification date of a representation, file or bitstream object, only a creation date for the new object. The act of modification (e.g., migration, normalization) is documented as an event and is linked to the object that is created as a result of these processes. In PREMIS, modification date is considered in the context of an event record that is associated with an Object, rather than a date associated with a history of changes to the metadata associated with an Object.

It is worth noting that most operating systems will have a different way of recording this information, and will use the term “last modified date” for the date (and time) that the current filewas created. Some operating systems also record a "creation date” which records the first date (and time) that a file of that name existed even though this is a different object (albeit often one that the current object was based off when it was created).

Object type: There is no semantic unit for a genre or media type that would classify objects on a much higher level than format (e.g., images or videos). The METS schema allows such expressivity with the presence of a USE attribute

that can be applied to files and file groups, but currently there is no controlled vocabulary defined for its values. This is potentially useful information to know at the system level (for example, for performing preservation actions on an entire class of materials) and possibly for categorizing objects in terms of how they are rendered in certain environments. However, high-level object typing is probably more useful for exchange and access to objects than for preservation purposes. Developing a universally acceptable list of object types is beyond the scope of PREMIS and, without an authority list of types, this element would not be entirely useful outside of the repository. This element might be recorded in descriptive metadata.

Permanence levels: The National Library of Medicine's Permanence ratings [39] appear to be less a property of an Object entity than a property of an entity defining business rules. Business rules are out of scope of PREMIS.

Profile conformance: A "profile" can be seen as a subtype or refinement of a format; for example, the GeoTIFF specification can be seen as a profile of TIFF. There was a question of whether profile conformance should be seen as something separate from format validation. The PREMIS *format* element is repeatable which means that both format conformance and a more specific profile conformance can be recorded. However, it is recommended to record only a single format at the most specific level (e.g., in the example conformance to GeoTIFF could be recorded which implies conformance to TIFF).

Reason for creation: This is really descriptive metadata and thus out of the scope of PREMIS. Also, PREMIS does not consider events or processes that occur before ingest to be core knowledge for a preservation repository. Some of the context surrounding object creation may be documented in relation to the Object entity in *creatingApplication* and the reason for creation could be recorded as part of the *eventDetail* for the Event of creation.

Sibling relationships: Sibling relationships always have a structural relationship (and may possibly also have a derivation relationship), and should therefore fall under these relationship categories rather than being a separate category of relationship. What renders them potentially confusing is that the parent is not always stored within the repository system. For example, a report created using Microsoft Word might be processed to create a PDF version for printing and an HTML version for online display. If both of these representations were stored in the preservation archive without the original Word file, it might not be obvious that the two representations have a sibling relationship.

39 National Library of Medicine, Developing Permanence Levels and the Archives for NLM's Permanent Web Documents, November 2007 (last updated 10 June 2010), http://www.nlm.nih.gov/psd/pcm/devpermanence.html.

GLOSSARY

GLOSSARY

This glossary defines a number of terms used in this Data Dictionary; recognizing thatin some cases other groups may have given different meanings to some of these terms. Terms were selected for inclusion in the glossary on the basis of their relative importance or frequency of occurrence in the report and Data Dictionary, and/or the potential for ambiguity or confusion in their interpretation.

Terms that are capitalized are defined elsewhere in the glossary.

Actionable:
Property of a Semantic Unit indicating that the Semantic Unit is recorded/ coded in such a way as to be processed automatically.

Agent:
Actor (human, machine, or software) associated with one or more Events and or Rights associated with a Digital Object. Machine or software Agents can themselves be captured as a Digital Object.

Anomaly:
Aspect of a Digital Object that does not meet the specification for the Digital Object.

Authenticity:
Property that a Digital Object is what it purports to be; that is, that the integrity of both the source and the content of the Digital Object can be verified.

Bit-Level Preservation:
Preservation strategy in which the sole objective is to ensure that a Digital Object remains fixed (unaltered) and viable (readable from media). No effort is made to ensure that the Digital Object remains renderable or interpretable by contemporary technology.

Bitstream:
Contiguous or non-contiguous data within a File that has meaningful common properties for preservation purposes. A Bitstream cannot be transformed into a standalone Filewithout the addition of File structure (headers, etc.) and/or reformatting the Bitstream in order to comply with some particular Format. Note that this definition is more specific than the common definition of “bitstream” used in computer science.

Business Rules:

Policies and other restrictions, guidelines, and procedures governing the administration and operation of a Preservation Repository.

Byte:
A component in the machine data hierarchy usually larger than a bit and smaller than a word: now most often eight bits and the smallest addressable unit of storage. A Byte typically holds one character. (From FOLDOC: http://foldoc.org/byte.)

Capture:
Process by which a Preservation Repository actively obtains Digital Objects for long-term retention, for example, a harvesting program that collects Web Sites. Note that the Capture process precedes the Ingest process.

Checksum:
See Message Digest.

Complex Object:
See Compound Object.

Compound Object:
Digital Object composed of multiple Files: for example, a Web of text and image Files

Compression:
Process of coding data to save storage space or transmission time. Although data is already coded in digital form for computer processing, it can often be coded more efficiently (using fewer bits). For example, run-lengthencoding replaces strings of repeated characters (or other units of data) with a single character and a count. There are many Compression algorithms and utilities. Compressed data must be Decompressed before it can be used. (From FOLDOC: http://foldoc.org/compression.)

Container:
In the Data Dictionary, a Semantic Unit used to group other related Semantic Units. A Container Semantic Unit takes no value of its own.

Core Preservation Metadata:
Semantic Units that most Preservation Repositories will need to know in order to support the digital preservation process. Core Preservation Metadata should be independent of factors such as specific preservation strategy, type of archived content, and institutional context.

Data File:
See File.

Data Object:

See Digital Object.

Deaccession:
Process of removing a Digital Object from the inventory of a Preservation Repository.

Deaccession:
Process of removing a Digital Object from the inventory of a Preservation Repository.

Decompression:
Process of reversing the effects of data Compression. From FOLDOC: http://foldoc.org/decompress.)

Decryption:
Process of employing any procedure used in cryptography to convert ciphertext (encrypted data) into plaintext. (From FOLDOC: http://foldoc.org/decryption.)

Deletion:
Process of removing a Digital Object from repository storage.

Dependency Relationship:
Relationship where one Digital Object requires another Object to support its function, delivery, or the coherence of its content.

Derivation Relationship:
Relationship between Digital Objects where one Object is the result of a Transformation performed on the other Object.

Descriptive Metadata:
Metadata that serves the purposes of discovery (how one finds a resource), identification (how a resource can be distinguished from other, similar resources), and selection (how to determine that a resource fills a particular need). (From Caplan, Metadata Fundamentals for All Librarians, ALA Editions, 2003

Digital Migration:
See Migration.

Digital Object:
Discrete unit of information in digital form. A Digital Object can be an Intellectual Entity, Representation, File, Bitstream, or Filestream. Note that the PREMIS definition of Digital Object differs from the definition commonly used in the digital library community, which holds a digital object to be a combination of identifier, metadata, and data.

Digital Provenance:

Documentation of processes in a digital Object's life cycle. Digital Provenance typically describes Agents responsible for the custody and stewardship of Digital Objects, key Events that occur over the course of the Digital Object's life cycle, and other information associated with the Digital Object's creation, management, and preservation.

Digital Provenance:
Documentation of processes in a digital Object's life cycle. Digital Provenance typically describes Agents responsible for the custody and stewardship of Digital Objects, key Events that occur over the course of the Digital Object's life cycle, and other information associated with the Digital Object's creation, management, and preservation.

Digital Provenance:
Documentation of processes in a digital Object's life cycle. Digital Provenance typically describes Agents responsible for the custody and stewardship of Digital Objects, key Events that occur over the course of the Digital Object's life cycle, and other information associated with the Digital Object's creation, management, and preservation.

Digital Signature:
Value computed with a cryptographic algorithm and appended to data in such a way that any recipient of the data can use the signature to verify the data's origin and integrity. The electronic counterpart of a handwritten signature on a hard copy document. From BBN Technologies: http://www.bbn.com/utility/glossary/D.)

Digital Signature Validation:
Process of determining that a decrypted digital signature matches an expected value when the correct keys, algorithms, and parameters have been used. Validationconfirms the originator and Fixity of the signed Digital Object.

Dissemination:
Process of retrieving a Digital Object from the Preservation Repository's archival storage and making it available to users. In the context of OAIS, Dissemination involves transforming one or more Archival Information Packages (AIP) into a Dissemination Information Package (DIP) and making it available in a form suitable for the Preservation Repository's Designated Community.

Dissemination:
Process of retrieving a Digital Object from the Preservation Repository's archival storage and making it available to users. In the context of OAIS, Dissemination involves transforming one or more Archival Information Packages (AIP) into a Dissemination Information Package (DIP) and making it available in a form suitable for the Preservation Repository's Designated Community.

Emulation:
Preservation strategy for overcoming technological obsolescence of hardware and software by developing techniques for imitating obsolete systems on future generations of computers. (From DPC: http://www.dpconline.org/advice/preservationhandbook/introduction/definitions-and-concepts?q=definitions.)

Encryption:
Process of employing any procedure used in cryptography to convert plaintext into ciphertext (encrypted message) in order to prevent any but the intended recipient from reading that data. Schematically, there are two classes of encryption primitives: public-key cryptography and private-key cryptography; they are generally used complementarily. Public-key encryption algorithms include RSA; private-key algorithms include the obsolescent Data Encryption Standard, the Advanced Encryption Standard, as well as RC4. (From FOLDOC: http://foldoc.org/encryption.)

Environment:
Technology supporting a Digital Object in some way (e.g. by rendering or executing it). Environments can consist of software, hardware, or a combination of both. Environments can be described as Intellectual Entities and captured and preserved in the Preservation Repository as Representations, Files and/or Bitstreams.

Entity:
Abstraction for a set of “things” (environments, Events, etc.) described by the same properties. The PREMIS data model defines four types of Entities: Objects, Agents, Rights, and Events.

Event:
Preservation-relevant action that involves at least one Digital Object and/or Agent known to the Preservation Repository.

Extensibility:
Property that Semantic Units in the PREMIS Data Dictionary may be supplemented by externally defined Semantic Units, or replaced by more granular Semantic Units, so long as there is no conflict in their definition and use.

Extensibility:
Property that Semantic Units in the PREMIS Data Dictionary may be supplemented by externally defined Semantic Units, or replaced by more granular Semantic Units, so long as there is no conflict in their definition and use.

File:

Named and ordered sequence of Bytes that is known by an operating system. A File can be zero or more Bytes long, has access permissions, and has File system statistics such as size and last modification date. A File also has a Format.

Filestream:
Embedded Bitstream that can be transformed into a standalone File without adding any additional information: for example, a TIFF image embedded within a tar File, or an encoded EPS within an XML File.

Fixity:
Property of a Digital Object that indicates it has not changed between two points in time.

Fixity Check:
Process of verifying that a File or Bitstream has not been changed during a given period. A common Fixity Check method is to compute a Message Digest ("hash") at one point and recalculate the Message Digest at a later point; if the digests are identical, the object has not been altered.

Format:
Specific, pre-established structure for the organization of a File, Bitstream, or Filestream.

Format Migration:
See Migration.

Forward Migration:
See Migration.

Granularity:
Relative size, scale, level of detail, or depth of penetration that characterizes an object or activity. "Level of granularity" may be used to refer to the level of focus in a hierarchy or to refer to the level of specificity of description.

Ingest:
Process of adding objects to a Preservation Repository's storage system. In the context of OAIS, Ingest includes services and functions that accept Submission Information Packages (SIP) from producers, and transform them into one or more Archival Information Packages (AIP) for long-term retention.

Ingest:
Process of adding objects to a Preservation Repository's storage system. In the context of OAIS, Ingest includes services and functions that accept Submission Information Packages (SIP) from producers, and transform them into one or more Archival Information Packages (AIP) for long-term retention.

Inhibitor:

Feature of a Digital Object intended to inhibit access, copying, Dissemination, or Migration. Common Inhibitors are Encryption and password protection.

Intellectual Entity:
Coherent set of content that is described as a unit: for example, a book, a map, a photograph, a serial. An Intellectual Entity can include other Intellectual Entities; for example, a Web Site can include a Web Page, a Web Page can include a photograph. An Intellectual Entity may have one or more Representations.

Media Migration:
Form of Replication, in which a Digital Object is copied onto a different type of digital storage medium because the original medium is in danger of obsolescence.

Media Migration:
Form of Replication, in which a Digital Object is copied onto a different type of digital storage medium because the original medium is in danger of obsolescence.

Media Refreshment:
Form of Replication, in which a Digital Object is copied onto a different unit of storage of the same or similar medium as the original. Note: Media Refreshment is used in preference to the definition of “refreshment” in the OAIS Reference Model. OAIS defines refreshment s a “Digital Migration where the effect is to replace a media instance with a copy that is sufficiently exact that all Archival Storage hardware and software continues to run as before.

Message Digest:
Result of applying a one-way hash function to a message. A Message Digest is a value that is shorter than the message, but would be different if the message were changed by even one character. (From BBN Technologies: http://www.bbn.com/utility/glossary/M.) “Message” here means any string of bits, such as a File or Bitstream. A Message Digest is often informally called a “checksum”.

Message Digest Calculation:
Process by which a Message Digest is created for a Digital Objectresiding in a Preservation Repository. See also Fixity Check.

Migration:
Preservation strategy in which a Transformation creates a version of a Digital Objectin a different Format, where the new Format is compatible with contemporary software and hardware environments. Ideally, Migration is accomplished with as little loss of content, formatting and functionality as possible, but the amount of information loss will vary depending on the

Formats and content types involved. Also called “format migration” and “forward migration.”

Note:
Migration and Media Migration are used in preference to the definition of “digital migration” in the OAIS Reference Model. OAIS defines digital migration as the “transfer of digital information, while intending to preserve it, within the OAIS. It is distinguished from transfers in general by three attributes: 1) a focus on the preservation of the full information content; 2) a perspective that the new archival implementation of the information is a replacement for the old; and 3) an understanding that full control and responsibility over all aspects of the transfer resides with the OAIS.”

Namespace:
Set of names in which all names are unique. From FOLDOC: http://foldoc.org/namespace.)

Normalization:
Form of Migration in which a version of a Digital Object is created in a new Format with properties more conducive to preservation treatment. Normalization is often implemented as part of the Ingest process.

Object:
See Digital Object.

Permission:
Agreement between a Rights holder and a Preservation Repository, allowing the Preservation Repository to undertake some action.

Pre-Ingest:
Period in the life cycle of a Digital Object before it is Ingested into a Preservation Repository.

Preservation Metadata:
Information a Preservation Repository uses to support the digital preservation process.

Preservation Repository:
Repository that, either as its sole responsibility or as one of multiple responsibilities, undertakes the long-term preservation of the Digital Objects in its custody.

Profile:
Specification for a particular implementation of a Format. For example, GeoTIFF is a profile of TIFF.

Quirk:

Any loss in functionality or change in the look and feel of a Digital Object resulting from the preservation processes and procedures implemented by a Preservation Repository. (See also the definition supplied by the National Library of Australia: http://www.nla.gov.au/preserve/pmeta.html#14.)

Refreshment:
See Media Refreshment.

Relationship:
Statement about an association between instances of Entities.

Render:
To make a Digital Object perceptible to a user, by displaying (for visual materials), playing (for audio materials), or other means appropriate to the Format of the Digital Object.

Replication:
Process of copying a Digital Object so that the copy is bit-wise identical to the original. Media Migration and Media Refreshment are specific types of Replication.

Representation:
Digital Object or physical object instantiating or embodying an Intellectual Entity. A Representation that is a Digital Object is the set of stored Files and Structural Metadataneeded to provide a complete rendition of the Intellectual Entity.

Rights:
Assertions of one or more legal entitlements or permissions pertaining to a Digital Object and/or an Agent.

Root:
The File that must be processed first in order to render a Representation correctly.

Semantic Component:
Semantic Unit grouped with one or more other Semantic Units within a Container. A Semantic Component may itself be a Container.

Semantic Unit:
Property of an Entity. Note: The PREMIS Data Dictionary makes a distinction between a Semantic Unit and a metadata element. A Semantic Unit is information that a Preservation Repository needs to know; a metadata element is how that information is actually recorded. So in practice there could be a one-to-one relationship between a Semantic Unit and its associated metadata element; a one-to-many relationship; or even a many-to-one relationship.

Ultimately, the translation of a set of Semantic Units into a corresponding set of metadata elements is an implementation issue.

Simple Object:
Digital Object consisting of a single File, for example, a technical report complete in one PDF File.

Store:
Write a File to some non-volatile storage device such as disk, tape, or DVD.

Structural Metadata:
Describes the internal structure of digital resources and the relationships between their parts. It is used to enable navigation and presentation. (From NINCH Guide to Good Practice: http://www.nyu.edu/its/humanities/ninchguide/appendices/metadata.html.)

Structural Relationship:
Relationship between parts of a Digital Object.

Technical Metadata:
Information describing physical (as opposed to intellectual) attributes or properties of Digital Objects. Some Technical Metadata properties are Format specific (that is, they pertain only to Digital Objects in a particular Format, for example, color space associated with a TIFF image), while others are Format independent (that is, they pertain to all Digital Objects regardless of Format, for example, size in bytes).

Transformation:
Process performed on a Digital Object that results in one or more new Digital Objects that are not bit-wise identical to the source Digital Object. Examples of Transformationinclude Migration and Normalization.

Validation:
Process of comparing a Digital Object with a standard or benchmark and noting compliance or exceptions. For example, a File can be validated against a File format specification or profile; a Representation can be validated against criteria for completeness.

Viability:
Property of being readable from media.

Virus Check:
Process of scanning a File for malicious programs designed to corrupt Digital Objects and systems.

Web Page:

A type of content found on the World Wide Web, usually in HTML/XHTML format (the File extensions are typically .htm or .html) and with hypertext links to enable navigation from one page or section to another. Web Pages often use associated graphics Files to provide illustration, and these too can be clickable links. (From Wikipedia: http://en.wikipedia.org/wiki/Web_page)

Web Site:
A collection of interlinked Web Pages held in the same location on the Internet, that is, HTML/XHTML documents accessible via HTTP on the internet; the set of all publicly accessible Web Sites in existence comprise the World Wide Web. The pages of a Web Site will be accessed from a common root URL, the home page, and usually reside on the same physical server. The URLs of the pages organize them into a hierarchy, although the hyperlinks between them control how the reader perceives the overall structure and how the traffic flows between the different parts of the Web Site. (From Wikipedia: http://en.wikipedia.org/wiki/Web_page).