

PREMIS Data Dictionary

for Preservation Metadata

version 2.0

March 2008



Contents:

Acknowledgments

Introduction

Background

The PREMIS Data Model

General Topics

Implementation Considerations

The PREMIS Data Dictionary Version 2.0

Special Topics

Methodology

Glossary

PREMIS Data Dictionary for Preservation Metadata

version 2.0

PREMIS Editorial Committee

March 2008

CONTENTS

Acknowledgments	ii
PREMIS Web Sites and E-Mail	iv
Introduction	1
Background	1
Development of the original PREMIS Data Dictionary	2
Implementable, core preservation metadata	3
The PREMIS Data Model	5
More on Objects	7
Intellectual Entities and Objects	9
More on Events	10
More on Agents	11
More on Rights	11
General Topics on the Structure and Use of the Data Dictionary	12
Identifiers	12
Relationships between Objects	13
Relationships between entities of different types	14
The 1:1 principle	14
Implementation Considerations	15
PREMIS conformance	15
Implementation of the data model	16
Storing metadata	17
Supplying metadata values	17
Extensibility	19
The PREMIS Data Dictionary Version 2.0	22
Limits to the scope of the Data Dictionary	23
Object Entity	25
Event Entity	130
Agent Entity	151
Rights Entity	157
Special Topics	195
Format information	195
Environment	197
Object characteristics and composition level: the “onion” model	199
Fixity, integrity, authenticity	200
Digital signatures	201
Non-core metadata	204
Methodology	208
Glossary	209

ACKNOWLEDGMENTS

PREMIS Editorial Committee members

Rebecca Guenther, Library of Congress, *Chair*
Steve Bordwell, General Register Office for Scotland
Olaf Brandt, Koninklijke Bibliotheek, Netherlands
Priscilla Caplan, Florida Center for Library Automation
Gerard Clifton, National Library of Australia
Angela Dappert, British Library
Markus Enders, Staats- und Universitätsbibliothek Göttingen/British Library
Brian Lavoie, OCLC
Bill Leonard, Library and Archives Canada
Zhiwu Xie, Los Alamos National Laboratory

Special thanks

These individuals contributed their expertise as former members of the PREMIS Editorial Committee:

Rory McLeod, British Library
Yaniv Levi, ExLibris

These individuals 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 WEB SITES AND E-MAIL

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

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

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

INTRODUCTION

Background

In June 2003, OCLC and RLG jointly sponsored the formation of the PREMIS (*Preservation Metadata: Implementation Strategies*) 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*.

That 237-page *Report* provides a wealth of resources on preservation metadata. First and foremost is the Data Dictionary itself, a comprehensive, practical resource for implementing preservation metadata in digital archiving 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.

In addition to the Data Dictionary, the working group also published a set of XML schema to support implementation of the Data Dictionary in digital archiving systems. The PREMIS Data Dictionary was awarded the 2005 Digital Preservation Award, given under the auspices of the British Conservation Awards, as well as the 2006 Society of American Archivists Preservation Publication Award.

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.

INTRODUCTION

The membership of the Editorial Committee reflects a variety of countries and institutional backgrounds.

At the time of the Data Dictionary's release, the decision was made to "freeze" its content for at least 18 months, giving the digital preservation community time to read and digest it, experiment with its implementation, identify errors, and most importantly, provide feedback on ways that the Data Dictionary could be improved to increase its value and ease of application. Feedback was collected through a variety of mechanisms, and in 2007, the Editorial Committee determined that a sufficient level of commentary had accumulated to warrant undertaking the first revision of the Data Dictionary. The members of the Editorial Committee revised the Data Dictionary, making every effort to engage stakeholders in the process of revision. The Committee kept the preservation community informed of issues being discussed, solicited comment on proposed revisions, and consulted outside experts where appropriate. The result of this process is the *PREMIS Data Dictionary for Preservation Metadata version 2.0*.

Development of the original PREMIS Data Dictionary

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. The survey provided an opportunity to explore the state of the art in digital preservation generally, and questions were drafted to elicit information about policies, governance and funding, system architecture, and preservation strategies, as well as metadata practices. The subgroup contacted 16 of 48 respondents by telephone for more in-depth interviews. In December 2004 the PREMIS working group published its report based on the survey of digital repositories, *Implementing Preservation Repositories 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.

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.

Implementable, core preservation metadata

The PREMIS 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.

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 intervention to supply or analyze. For example, coded 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

INTRODUCTION

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.

The PREMIS Data Model

The working group developed a simple data model to organize the semantic units defined in the Data Dictionary. The data model defines five entities the working group felt were particularly important in regard to digital preservation activities: Intellectual Entities, Objects, Events, Rights, and Agents.⁴ Each semantic unit defined in the Data Dictionary is a property of one of the entities in the data model. [Figure 1](#) provides a graphical illustration of the PREMIS Data Model.

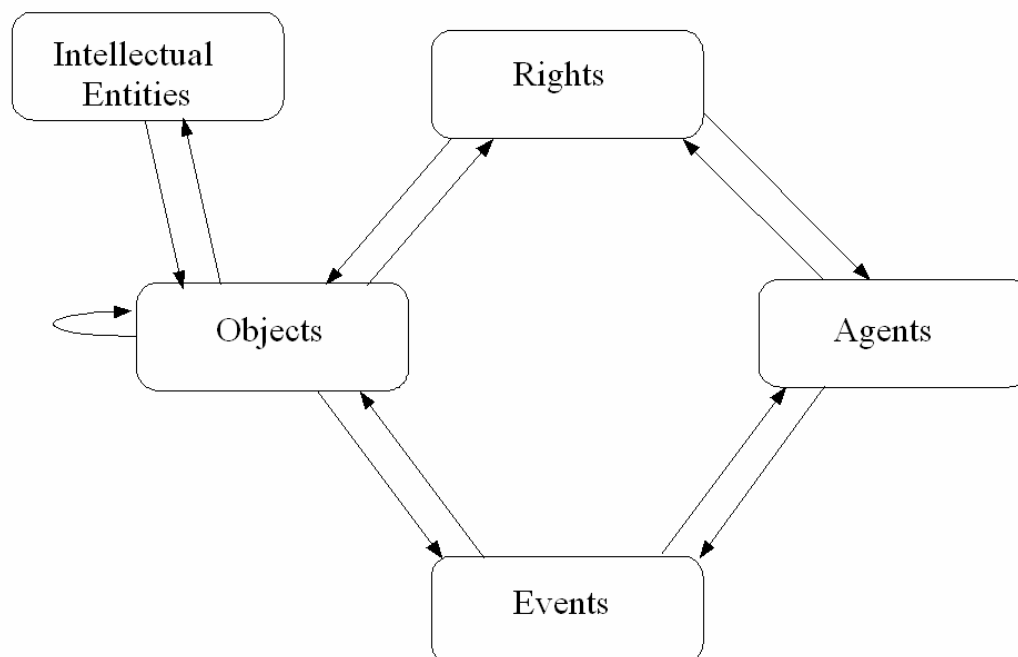


Figure 1: The PREMIS Data Model

In [Figure 1](#), entities are represented by boxes; relationships between entities are represented by arrows. The direction of the arrow indicates the direction of the relationship linkage as it is recorded in the preservation metadata. For example, the arrow pointing from the Rights entity to the Agents entity means that the metadata associated with the Rights entity includes a semantic unit recording information about the relationship with an Agent.

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.

INTRODUCTION

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

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

Object (or Digital Object): a discrete unit of information in digital form.⁵

Event: an action that involves or impacts 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.

Rights: assertions of one or more rights or permissions pertaining to an Object and/or Agent.

The PREMIS Data Dictionary defines **semantic units**. 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. If a migration event creates 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 semantic unit is 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.

In some cases a semantic unit takes the form of a **container** that groups a set of related semantic units. For example, a 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 three subtypes: file, bitstream, and representation.

A **file** is a named and ordered sequence of bytes that is known by 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.

A **representation** is the set of files, including structural metadata, needed for a complete and reasonable 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.

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 properties different from the file itself. However, in another example, three TIFF files could be aggregated within a larger tar file. In this case the three TIFF files are also embedded bitstreams, but they have all the properties of TIFF files.

The PREMIS data model refines the definition of bitstream to include only an embedded bitstream that cannot be transformed into a standalone file 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.

INTRODUCTION

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.

Representations

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

PREMIS chose the term “representation” to avoid the term “manifestation” as it is used in the *Functional Requirements for Bibliographic Records* (FRBR).⁶ In FRBR a manifestation entity is “all the physical objects that bear the same characteristics in respect to both intellectual content and physical form.” In the PREMIS model a representation is *a single digital instance of an intellectual entity held in a preservation repository*.

A preservation repository might hold more than one representation for 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 JPEG2000 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 files would 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 chose to store a single copy of the SGML file, which would then be shared between representations. Alternately, the repository could choose to duplicate the SGML file and store two identical copies of it. The two representations would then consist of the TIFF and SGML copy 1, and the JPEG2000 and SGML copy 2.

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 the repository does not manage representations, it does not need to record metadata about them.

Intellectual Entities and Objects

The relationship between Intellectual Entities and Objects can be illustrated by a couple of examples:

Example 1, *Animal Antics*: 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. 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 functionalities. The repository will record metadata about two representation objects and 191 file objects.

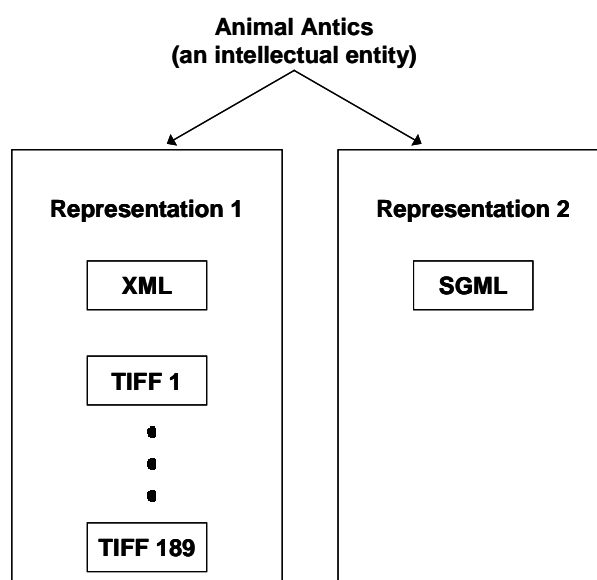


Figure 2: *Animal Antics* Intellectual Entity Example

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 representation object, one file object, and two bitstream objects.

INTRODUCTION

More on Events

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

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 *relatedEventIdentification*. 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 page [13](#).)

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 = "derived from"
relatedObjectIdentification
  relatedObjectIdentifierType = "local"
  relatedObjectIdentifierValue = "A"
  relatedObjectSequence = "not applicable"
relatedEventIdentification
  relatedEventIdentifierType = "local"
  relatedEventIdentifierValue = "1"
  relatedEventSequence = "not applicable"
```

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.

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.

The data model diagram shows an arrow from the Agent entity to the Event entity, but no arrow from Agent to the Object entity. Agents influence Objects only indirectly through Events. Each Event can have one or more related Objects and 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.

More on Rights

Many efforts are concerned with metadata related to intellectual property rights and permissions, from rights expression languages to the <indecs> 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 Coyle⁷. 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 this version. In this revision the Editorial Committee relied heavily upon the Coyle paper, background materials such as Peter Hirtle's excellent “Digital Preservation and Copyright,”⁸ and the California Digital Library's draft copyrightMD schema⁹. 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.

INTRODUCTION

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 repository system, it is unlikely that 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, identifier type should be explicitly supplied.

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. The working group began its exploration of this topic by collecting examples from existing preservation metadata projects. It found a wide range of metadata facts 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”). The group also reviewed the element refinements for the Dublin Core Relation element (IsPartOf, IsFormatOf, IsVersionOf, etc.) and concluded that most relationships among objects appear to be variants of these three basic types: structural, derivation, and dependency.

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., a photograph) 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

INTRODUCTION

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. Rather than specify a particular approach, the group identified essential information that must be captured. The PREMIS Data Dictionary describes this in the semantic 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 METS¹⁰ structMap or descriptive metadata schemes that define relationship types (e.g. Dublin Core¹¹).

A **dependency relationship** exists when one object requires another to support its function, delivery, or coherence of content. An object may require a font, style sheet, DTD, schema, or other file that is not formally part of the object itself but is necessary to render it. The Data Dictionary handles dependency relationships as part of the environment information, in the semantic units *dependency* and *swDependency*. 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 of the object.

Relationships between entities of different types

The data model diagram uses arrows to show relationships between entities of different types. Objects are related to Intellectual Entities, 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 for use as a pointer. So, for example, the Object entity has arrows pointing to Intellectual Entities and Events. These are implemented in the Data Dictionary by the semantic units *linkingIntellectualEntityIdentifier* and *linkingEventIdentifier*.

The 1:1 principle

In digital preservation it is common practice to create new copies or versions of stored objects. For example, in forward 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, every Object held within the preservation repository (file, bitstream, representation) is 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. 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 *relationship* semantic unit associated with the Object entity. The semantic component *relatedEventIdentification* should be used to make the association with the Event.

Implementation Considerations

PREMIS conformance

PREMIS conformance requires a preservation repository to follow the specifications outlined in the Data Dictionary. For example, if the repository claiming to be PREMIS-conformant implements a metadata element sharing the name of a semantic unit in the Data Dictionary, it is expected that the repository's metadata element will also share the definition of the semantic unit. Metadata not defined in the Data Dictionary may certainly be used, but non-PREMIS elements should not conflict with or overlap with PREMIS semantic units if they use the same names. Data constraints and applicability guidelines in the Data Dictionary must also be adhered to. For repeatability and obligation, PREMIS conformance permits more stringent but not more liberal application. That is, a semantic unit defined in the Data Dictionary as repeatable can be treated as not repeatable within a repository, but not vice versa.

The PREMIS Data Dictionary designates some semantic units as mandatory when describing representations, files, and/or bitstreams. The mandatory semantic units represent the minimum amount of information 1) necessary to support the long-term preservation of digital objects, and 2) that must accompany a digital object as it is transferred from the custody of one preservation repository to another. There is no prescribed strategy for collecting, storing, or managing the mandatory semantic units within the repository's internal systems. Nor is there a minimum level of information that must be explicitly recorded and maintained locally by the repository. In general, the mandatory semantic units of the Data Dictionary represent the information that a preservation repository must be able to associate with any archived digital object in its possession. The specific means of association (e.g., local metadata storage, shared registries, etc.) are implementation issues and outside the scope of the Data Dictionary.

When a digital object is exchanged between two preservation repositories, the repository sending the object must be able to extract from its systems or from other sources the information needed to populate the semantic units marked mandatory in the Data Dictionary. This information must conform to the specifications in the Data Dictionary and must be packaged with the digital object before its transfer to the second repository. The PREMIS working group believes that this information represents the minimum amount for the second repository to accept custody of the digital object and assume responsibility for its long-term preservation.

Some PREMIS semantic units are equivalent to metadata elements in other metadata schemas. If metadata is taken from other schemas to populate PREMIS semantic units, care must be taken to ensure that this information conforms to the requirements and constraints associated with the

INTRODUCTION

corresponding semantic unit in the PREMIS Data Dictionary. Harmonizing the PREMIS Data Dictionary with other metadata schemas in cases where they overlap would help minimize conformance issues. For example, the Z39.87 metadata standard (Technical Metadata for Digital Still Images)¹² revised some of its elements to harmonize them with equivalent semantic units in the PREMIS Data Dictionary.

Sometimes a preservation repository exchanges digital objects with parties that are not themselves preservation repositories. When a party submits an object to a preservation repository for archival retention, it is unlikely that the submitter will be in a position to supply the full range of information needed to populate the mandatory semantic units. Instead, it will supply a subset of this information whose extent, ideally, is determined by prior arrangement between the submitter and the repository. Whatever the extent of this subset, any information supplied by the submitter should conform to the Data Dictionary. The repository's ingest process would then supply the rest of the information for the mandatory semantic units.

When a repository disseminates an archived digital object to a user, it is unlikely that the user will be interested in the full range of mandatory semantic units associated with the archived object. Instead, the user would be provided with a subset of these semantic units. As in the case of submission, whatever the extent of this subset, any information supplied by the repository should conform to the Data Dictionary.

Achieving interoperability across a network of preservation repositories and other stakeholders requires a shared view of the metadata needed to support long-term preservation, formalized as an implementable schema. PREMIS conformance and the mandatory semantic units are intended to fill this need.

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 subclasses of objects, such as files and filestreams, bitstreams, and representations. 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, *environment* is a property of Objects. Logically, each file has one or more associated environments. However, 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. 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 set of environment

information. The “file” table can be joined with the “environment” table to get the appropriate environment information for each file.

- Repository 2 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.
- Repository 3 uses a system that 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 a format property. Because format description is also an object, it could include an environment property, which in turn would point to an environment description object. Alternatively, a file object could include an environment property directly.

Storing metadata

The survey by the Implementation Strategies Subgroup showed that repositories have implemented several different architectures for storing metadata. Most commonly, metadata is stored in relational database tables. It is also common to store metadata as XML documents in an XML database, or as XML documents stored with the content data files. Other methods include proprietary flat file formats and object-oriented databases. Most respondents were using two or more of these methods. (For more information, see the Implementation Survey Report².)

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 “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 METS¹⁰, MPEG-21¹³, or SMIL¹⁴. 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.

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

INTRODUCTION

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, the group tended 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 filename includes a file type 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 it does in some cases indicate suggested values.

The PREMIS Editorial Committee concluded that implementers should be able to choose the vocabulary used and specify which vocabulary is used. Whether and how to validate that the appropriate values have been used is an implementation consideration. With version 2.0 of the PREMIS Data Dictionary, the PREMIS Maintenance Activity at the Library of Congress is establishing a mechanism to register controlled vocabularies in use with PREMIS semantic units and expose them in a way that the PREMIS schemas can include them. 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, METS¹⁰ users may specify controlled vocabularies used in metadata in a METS profile, or PREMIS profiles may be established to document the same. A mechanism to record the source is provided in the PREMIS XML schemas. Other XML implementations may develop mechanisms to declare controlled vocabularies used or to validate values against specified vocabularies.

In Resource Description Framework (RDF), use of resource URIs as property values is encouraged, and many XML Schemas require attribute values to be URIs.¹⁵ 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/xmldsign#dsa-sha1”.

In general, resource URIs are allowable as values for semantic units in the PREMIS Data Dictionary, unless some noted constraint would disallow this. However, the working group was wary of recommending this practice for preservation. Resolution of URIs depends on a protocol that while currently ubiquitous is outside the control of the preservation repository. Also, the group felt strongly that any information needed for long-term preservation should be stored within the repository itself. If this information is stored as a preservation object, it is best referenced by the repository’s *objectIdentifier*. Information stored otherwise should still be under the direct control of the repository. Therefore, most examples in the Data Dictionary are names

of values rather than resource URIs. The equivalent of the example above might be simply “DSA-SHA1,” which should be assumed to be a constant whose meaning is known to the repository through some table or other documentation under the control of the repository organization.

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 introduces a formal mechanism for extensibility within the schemas for a small number of semantic units which were deemed prime candidates for extension. Later revisions of the Data Dictionary may add to this initial set of extensible semantic units if warranted.

The initial set of semantic units for which extensibility will be supported in the schemas is:

- `significantProperties` [Object entity]
- `objectCharacteristics` [Object entity]
- `creatingApplication` [within `objectCharacteristics`, Object entity]
- `environment` [within `objectCharacteristics`, Object entity]
- `signatureInformation` [Object entity]
- `eventOutcomeDetail` [within `eventOutcomeInformation`, Event entity]
- `rights` [Rights 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.

A new container semantic unit, *objectCharacteristicsExtension*, has also been created within the Object entity to allow inclusion of format specific technical metadata within PREMIS.

In devising the mechanism for extensibility, the PREMIS Editorial Committee adopted the principle 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) to enable extension of *eventOutcomeDetail*.

INTRODUCTION

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

- An **extension** container may be used to either supplement or replace PREMIS semantic units within the parent container (that is, the container which includes the extension container). The one exception is *objectCharacteristicsExtension*, which may only supplement *objectCharacteristics*.
- An **extension** container may be used with existing PREMIS semantic units, supplementing the PREMIS semantic units with additional metadata.
- An **extension** container may be used without existing PREMIS semantic units, effectively replacing the PREMIS semantic units with other applicable metadata (except for *objectCharacteristicsExtension*).
- 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 unit the parent container is repeated with appropriate subunit. If extensions from different external schemas are needed, the parent container should also 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.

Date and time formats in PREMIS

All semantic units that specify the use of a date or date and 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. Version 2.0 of 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 termOfGrant)
- endDate (under termOfGrant)

THE PREMIS DATA DICTIONARY VERSION 2.0

The PREMIS Data Dictionary includes semantic units for Objects, Events, Agents, and Rights. The fifth entity in the model, the Intellectual Entity, is considered out of scope because it is well served by descriptive metadata. 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 "[Special Topics](#)" page [195](#).

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 a representation, file, or bitstream Object. Semantic units that apply to files also apply to filestreams (see page [7](#)).
- **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 an 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. MARC¹⁶, MODS¹⁷, the Dublin Core Metadata Element Set¹¹, the Content Standard for Digital Geospatial Metadata¹⁸, the VRA Core¹⁹, the Encoded Archival Description (EAD)²⁰, and the Data Documentation Initiative²¹ schemas are only some of the standards that define descriptive metadata elements. The working

THE PREMIS DATA DICTIONARY

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 MARC¹⁶, vCard²², MADS²³, 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. This revision broadens the semantic units used for rights information and allows 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, 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 or hardware details: The working group did not attempt to define metadata for detailed documentation of media or hardware. 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 and hardware 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.

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 unit that applies to all categories of object (representation, file, and bitstream) is *objectIdentifier*.

Entity types

- **Representation:** A digital object instantiating or embodying an Intellectual Entity. A representation is the set of stored digital files and structural metadata needed to provide a complete and reasonable rendition of the Intellectual Entity.
- **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.

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) [representation, file]
 - 1.3.1 preservationLevelValue (M, NR) [representation, file]
 - 1.3.2 preservationLevelRole (O, NR) [representation, file]
 - 1.3.3 preservationLevelRationale (O, R) [representation, file]
 - 1.3.4 preservationLevelDateAssigned (O, NR) [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 (M, NR) [file, bitstream]

THE PREMIS DATA DICTIONARY

- 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) [representation, file]
- 1.7 storage (M, R) [file, bitstream]
 - 1.7.1 contentLocation (O, NR) [file, bitstream]
 - 1.7.1.1 contentLocationType (M, NR) [file, bitstream]
 - 1.7.1.2 contentLocationValue (M, NR) [file, bitstream]
 - 1.7.2 storageMedium (O, NR) [file, bitstream]
- 1.8 environment (O, R)
 - 1.8.1 environmentCharacteristic (O, NR)
 - 1.8.2 environmentPurpose (O, R)
 - 1.8.3 environmentNote (O, R)
 - 1.8.4 dependency (O, R)
 - 1.8.4.1 dependencyName (O, R)
 - 1.8.4.2 dependencyIdentifier (O, R)
 - 1.8.4.2.1 dependencyIdentifierType (M, NR)
 - 1.8.4.2.2 dependencyIdentifierValue (M, NR)
 - 1.8.5 software (O, R)
 - 1.8.5.1 swName (M, NR)

- 1.8.5.2 swVersion (O, NR)
- 1.8.5.3 swType (M, NR)
- 1.8.5.4 swOtherInformation (O, R)
- 1.8.5.5 swDependency (O, R)
- 1.8.6 hardware (O, R)
 - 1.8.6.1 hwName (M, NR)
 - 1.8.6.2 hwType (M, NR)
 - 1.8.6.3 hwOtherInformation (O, R)
- 1.8.7 environmentExtension (O, R)
- 1.9 signatureInformation (O, R) [file, bitstream]
 - 1.9.1 signature (O, R)
 - 1.9.1.1 signatureEncoding (M, NR) [file, bitstream]
 - 1.9.1.2 signer (O, NR) [file, bitstream]
 - 1.9.1.3 signatureMethod (M, NR) [file, bitstream]
 - 1.9.1.4 signatureValue (M, NR) [file, bitstream]
 - 1.9.1.5 signatureValidationRules (M, NR) [file, bitstream]
 - 1.9.1.6 signatureProperties (O, R) [file, bitstream]
 - 1.9.1.7 keyInformation (O, NR) [file, bitstream]
 - 1.9.2 signatureInformationExtension (O, R) [file, bitstream]
- 1.10 relationship (O, R)
 - 1.10.1 relationshipType (M, NR)
 - 1.10.2 relationshipSubType (M, NR)
 - 1.10.3 relatedObjectIdentification (M, R)
 - 1.10.3.1 relatedObjectIdentifierType (M, NR)
 - 1.10.3.2 relatedObjectIdentifierValue (M, NR)
 - 1.10.3.3 relatedObjectSequence (O, NR)
 - 1.10.4 relatedEventIdentification (O, R)
 - 1.10.4.1 relatedEventIdentifierType (M, NR)
 - 1.10.4.2 relatedEventIdentifierValue (M, NR)
 - 1.10.4.3 relatedEventSequence (O, NR)
- 1.11 linkingEventIdentifier (O, R)
 - 1.11.1 linkingEventIdentifierType (M, NR)
 - 1.11.2 linkingEventIdentifierValue (M, NR)
- 1.12 linkingIntellectualEntityIdentifier (O, R)
 - 1.12.1 linkingIntellectualEntityIdentifierType (M, NR)
 - 1.12.2 linkingIntellectualEntityIdentifierValue (M, NR)
- 1.13 linkingRightsStatementIdentifier (O, R)
 - 1.13.1 linkingRightsStatementIdentifierType (M, NR)
 - 1.13.2 linkingRightsStatementIdentifierValue (M, NR)

THE PREMIS DATA DICTIONARY

Semantic unit	1.1 objectIdentifier		
Semantic components	1.1.1 objectIdentifierType 1.1.2 objectIdentifierValue		
Definition	A designation used to uniquely identify the object within the preservation repository system in which it is stored.		
Rationale	Each data object held in the preservation repository must have a unique identifier to relate it to descriptive, technical, and other metadata.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Mandatory	Mandatory	Mandatory
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 automatically or manually generated. Recommended practice is for repositories to use identifiers 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 object to information held outside the repository.		
Usage notes	<p>The <i>objectIdentifier</i> is mandatory if the preservation repository stores and manages objects at that level (i.e., representation, file, bitstream).</p> <p>The <i>objectIdentifier</i> is repeatable in order to allow both repository-assigned and externally-assigned identifiers to be recorded. See Creation/Maintenance note above.</p> <p>Identifiers must be unique within the repository. They may be preexisting, and in use in other digital object management systems.</p> <p>Identifiers used to identify a class of objects (e.g., the way an ISBN identifies all books in the same edition) are not acceptable as identifiers in the context of the preservation repository, which must identify the specific object in the repository.</p> <p>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 explicitly recorded. Similarly, if the repository uses only one type of</p>		

	<p>identifier, the type can be assumed and does not need to be explicitly recorded.</p> <p>A persistent identifier should be used, but the particular identifier scheme is an implementation specific decision.</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

THE PREMIS DATA DICTIONARY

Semantic unit	1.1.1 objectIdentifierType		
Semantic components	None		
Definition	A designation of the domain within which the object identifier is unique.		
Rationale	Identifier values cannot be assumed to be unique across domains; the combination of <i>objectIdentifierType</i> and <i>objectIdentifierValue</i> should ensure uniqueness.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	DLC DRS hdl:4263537	DLC DRS hdl:4263537	DLC DRS hdl:4263537
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	The type of the identifier may be implicit within the repository as long it is can be explicitly communicated when the digital object is disseminated outside of it.		

Semantic unit	1.1.2 objectIdentifierValue		
Semantic components	None		
Definition	The value of the <i>objectIdentifier</i> .		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	0000000312	IU2440 WAC1943.56 AMNH CD269/CD269/70/10 596.PCD CDS-VDEP- 200211119- 24879.734 1001/dig/pres/2004- 024 http://nrs.harvard.edu /urn- 3:FHCL.Loeb:sa1	IU2440-1 IU2440-2
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	1.2 objectCategory		
Semantic components	None		
Definition	The category of object to which the metadata applies.		
Rationale	Preservation repositories are likely to treat different categories of objects (representations, files, and bitstreams) differently in terms of metadata and data management functions.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	representation	file	bitstream
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	Suggested values: representation, file, bitstream. A filestream should be considered a file.		

Semantic unit	1.3 preservationLevel		
Semantic components	1.3.1 preservationLevelValue 1.3.2 preservationLevelRole 1.3.3 preservationLevelRationale 1.3.4 preservationLevelDateAssigned		
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.		
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. The context surrounding the choice of a particular preservation option for an object may also require further explanation.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Repeatability	Repeatable	Repeatable	
Obligation	Optional	Optional	
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	<p>If the repository offers only a single preservation level, this value does not need to be explicitly recorded within the repository.</p> <p>Application of a particular set of <i>preservationLevel</i> semantic units may only cover a single representation of an object: representations in other technical forms or serving other functions may have a different <i>preservationLevel</i> applied.</p> <p>The container may be repeated if a preservation level value needs to be recorded in additional contexts (see <i>preservationLevelRole</i>, page 35).</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.3.1 preservationLevelValue		
Semantic components	None		
Definition	A value indicating the set of preservation functions expected to be applied to the object.		
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.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Examples	bit-level full 0 1 2	bit-level full 0 fully supported with future migrations	
Repeatability	Not repeatable	Not repeatable	
Obligation	Mandatory	Mandatory	
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 <i>preservationLevelValue</i> may be recorded per <i>preservationLevel</i> container. If a further <i>preservationLevelValue</i> applies to the object in a different context, a separate <i>preservationLevel</i> container should be repeated.		

Semantic unit	1.3.2 preservationLevelRole		
Semantic components	None		
Definition	A value indicating the context in which a set of preservation options is applicable.		
Rationale	Repositories may assign <i>preservationLevelValues</i> in different contexts which must be differentiated, and may need to record more than one context.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Examples	requirement intention capability	requirement intention capability	
Repeatability	Not repeatable	Not repeatable	
Obligation	Optional	Optional	
Usage notes	<p>This optional semantic unit qualifies the sense or context in which the <i>preservationLevelValue</i> in the current <i>preservationLevel</i> container is applied.</p> <p>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. <i>preservationLevelRole</i>=“requirement”) and the current capability (e.g. <i>preservationLevelRole</i>=“capability”).</p> <p>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 <i>preservationLevelValue</i> 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.</p> <p>It is good practice to specify <i>preservationLevelRole</i> for clarity even if the repository only assigns <i>preservationLevelValue</i> in one sense or context. If more than one <i>preservationLevel</i> is recorded, <i>preservationLevelRole</i> should always be supplied.</p>		

THE PREMIS DATA DICTIONARY

	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 <i>preservationLevel</i> containers should be used.
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Semantic unit	1.3.3 preservationLevelRationale		
Semantic components	None		
Definition	The reason a particular <i>preservationLevelValue</i> was applied to the object.		
Rationale	Application of a particular <i>preservationLevelValue</i> may require justification, especially if it differs from that usually applied according to repository policy.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Examples	user pays legislation	defective file bit-level preservation only available for this format	
Repeatability	Repeatable	Repeatable	
Obligation	Optional	Optional	
Usage notes	<p>This optional semantic unit records the reason for applying the <i>preservationLevelValue</i>.</p> <p>This information can be particularly important when the assigned <i>preservationLevelValue</i> differs from usual repository policy.</p> <p>For example, a repository may normally assign a <i>preservationLevelValue</i> 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 <i>preservationLevelValue</i> of “bit-level preservation” to this file, recording “defective file” as the rationale.</p> <p>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.</p> <p><i>preservationLevelRationale</i> may be repeated if more than one reason needs to be recorded.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.3.4 preservationLevelDateAssigned		
Semantic components	None		
Definition	The date, or date and time, when a particular <i>preservationLevelValue</i> was assigned to the object.		
Rationale	The <i>preservationLevel</i> 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 <i>preservationLevelValue</i> 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.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Examples	2007-11-05 2007-11-05T08:15:30-05:00 20080315	2007-11-05 2007-11-05T08:15:30-05:00 20080315	
Repeatability	Not repeatable	Not repeatable	
Obligation	Optional	Optional	

Semantic unit	1.4 significantProperties		
Semantic components	1.4.1 significantPropertiesType 1.4.2 significantPropertiesValue 1.4.3 significantPropertiesExtension		
Definition	Characteristics of a particular object subjectively determined to be important to maintain through preservation actions.		
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		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Creation / Maintenance notes	<p>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 repository.</p>		
Usage notes	<p>All of this semantic unit's subunits are optional. At least one of the <i>significantPropertiesValue</i> and <i>significantPropertiesExtension</i> subunits must be present if this container is included or both may be used.</p> <p>Significant properties may be objective technical characteristics subjectively considered important, or subjectively determined characteristics. For example, a PDF may contain links that are not considered important and JavaScript that is considered important. Or future migrations of a TIFF image may require optimization for line clarity or for color; the option chosen would depend upon a curatorial judgment of the significant properties of the image.</p> <p>Listing significant properties implies that the repository plans to preserve these properties across time and requires them to acceptably survive preservation action; for example, to be maintained during emulation or after format migration. It also implies that the repository would note when preservation action results in modification of significant properties.</p> <p>In practice, significant properties might be used as measures of</p>		

	<p>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.</p> <p>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.</p> <p>The semantic units included in the <i>significantProperties</i> 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.</p> <p>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 could include:</p> <p><i>significantPropertiesType</i> = “content”</p> <p><i>significantPropertiesValue</i> = “all textual content and images”</p> <p><i>significantPropertiesType</i> = “behavior”</p> <p><i>significantPropertiesValue</i> = “editable”</p> <p>Other repositories may choose to describe significant properties at a more granular attribute level; for example:</p> <p><i>significantPropertiesType</i> = “page count”</p> <p><i>significantPropertiesValue</i> = “7”</p> <p><i>significantPropertiesType</i> = “page width”</p> <p><i>significantPropertiesValue</i> = “210 mm”</p> <p>Each facet:detail pair should be contained in a separate, repeated <i>significantProperties</i> container.</p> <p>Further work on determining and describing significant properties may yield more detailed schemes to facilitate general description.</p> <p>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 significant properties volume and timing, which are recorded as <i>significantProperties</i> of A. In migrated version B, the timing is modified, which is noted in the <i>eventOutcome</i> of the migration event. Only volume is listed as a significant property of B.</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Semantic unit	1.4.1 significantPropertiesType		
Semantic components	None		
Definition	The aspect, facet, or attribute of an object about which significant properties are being described.		
Rationale	Repositories may choose to describe significant properties based on a particular aspect or attribute of an object.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
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
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Usage notes	This semantic unit is optional and may be used as part of a facet:detail pair with <i>significantPropertiesValue</i> .		

THE PREMIS DATA DICTIONARY

Semantic unit	1.4.2 significantPropertiesValue		
Semantic components	None		
Definition	Description of the characteristics of a particular object subjectively determined to be important to maintain through preservation actions.		
Rationale	Repositories may choose to describe significant properties based on a particular aspect or attribute of an object.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	<p>[for a Web page containing animation that is not considered essential] Content only.</p> <p>[For detail associated with a <i>significantProperties</i> Type of “behavior”] “hyperlinks traversable”</p>	<p>[for a word processed document with embedded links that are not considered essential] Content only.</p> <p>[For detail associated with a <i>significantProperties</i> Type of “behavior”] “editable”</p> <p>[For detail associated with a <i>significantProperties</i> Type of “page width”] 210 mm</p>	<p>[for a PDF with an embedded graph, where the lines’ color determines the lines’ meaning] Color.</p> <p>[For detail associated with a <i>significantProperties</i> Type of “appearance”] Color.</p>
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Usage notes	<p>If facet:detail pairs are used, the content of <i>significantPropertiesValue</i> should describe the significant properties of object relevant to the aspect, facet, or attribute declared in the <i>significantPropertiesType</i> with which it is paired.</p> <p>If facet:detail pairs are not used, <i>significantPropertiesValue</i> may be used to freely describe any characteristic of an object.</p> <p><i>significantPropertiesValue</i> is not repeatable. Multiple significant properties should be described in separate, repeated <i>significantProperties</i> container units.</p>		

Semantic unit	1.4.3 significantPropertiesExtension		
Semantic components	Defined externally		
Definition	A container to include semantic units defined outside of PREMIS for significant properties.		
Rationale	There may be a need to replace or extend PREMIS defined units.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	<p>All of this semantic unit's subunits are optional. At least one of the <i>significantPropertiesValue</i> and <i>significantPropertiesExtension</i> subunits must be present if this container is included.</p> <p>If the <i>significantPropertiesExtension</i> container needs to be associated explicitly with any PREMIS subunit under <i>significantProperties</i>, the container <i>significantProperties</i> is repeated. If extensions from different external schemas are needed, <i>significantProperties</i> should also be repeated.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.5 objectCharacteristics		
Semantic components	1.5.1 compositionLevel 1.5.2 fixity 1.5.3 size 1.5.4 format 1.5.5 creatingApplication 1.5.6 inhibitors 1.5.7 objectCharacteristicsExtension		
Definition	Technical properties of a file or bitstream that are applicable to all or most formats.		
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 <i>objectCharacteristicsExtension</i> .		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Mandatory	Mandatory
Usage notes	<p>The semantic units included in <i>objectCharacteristics</i> should be treated as a set of information that pertains to a single object at a single <i>compositionLevel</i>. Object characteristics may be repeated when an object was created by applying two or more encodings, such as compression and encryption. In this case each repetition of <i>objectCharacteristics</i> would have an incrementally higher <i>compositionLevel</i>.</p> <p>When encryption is applied, the <i>objectCharacteristics</i> block must include an inhibitors semantic unit.</p> <p>A bitstream embedded within a file may have different object characteristics than the file. Where these characteristics are relevant for preservation, they should be recorded.</p> <p>When a single file is equivalent to a representation, <i>objectCharacteristics</i> may be applied and thus associated with the representation. In these cases, the relationship between the file comprising the representation and other associated files may be expressed using <i>relationshipSubType</i> (see page 112).</p>		

Semantic unit	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.		
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		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		0 1 2	0 1 2
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
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	<p>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 unencrypt file C to create file B and then uncompress file B to create file A.</p> <p>A <i>compositionLevel</i> 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.</p> <p>Numbering goes lowest to highest (first encoded = 0). 0 is base object; 1-n are subsequent encodings.</p> <p>Use 0 as the default if there is only one <i>compositionLevel</i>.</p> <p>When multiple file objects are bundled together as filestreams within</p>		

THE PREMIS DATA DICTIONARY

	<p>a package file object (e.g., a ZIP file), the individual filestream objects are <i>not</i> 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,” page 199.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Semantic unit	1.5.2 fixity		
Semantic components	1.5.2.1 messageDigestAlgorithm 1.5.2.2 messageDigest 1.5.2.3 messageDigestOriginator		
Definition	Information used to verify whether an object has been altered in an undocumented or unauthorized way.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable (see usage note)	Applicable	Applicable (see usage note)
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Creation / Maintenance notes	Automatically calculated and recorded by repository.		
Usage notes	<p>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. Recommended practice is to use two or more message digests calculated by different algorithms. (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.)</p> <p>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 <i>eventOutcome</i>. Therefore, only the <i>messageDigestAlgorithm</i> and <i>messageDigest</i> need to be recorded as <i>objectCharacteristics</i> for future comparison.</p> <p>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.</p> <p>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</p>		

THE PREMIS DATA DICTIONARY

	<p>or SHA-1 message digests in internal metadata that was calculated on any range of bytes of the file.</p> <p>See “Fixity, integrity, authenticity,” page 200.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Semantic unit	1.5.2.1 messageDigestAlgorithm		
Semantic components	None		
Definition	The specific algorithm used to construct the message digest for the digital object.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		MD5 Adler-32 HAVAL SHA-1 SHA-256 SHA-384 SHA-512 TIGER WHIRLPOOL	
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.2.2 messageDigest		
Semantic components	None		
Definition	The output of the message digest algorithm.		
Rationale	This must be stored so that it can be compared in future fixity checks.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		7c9b35da4f2ebd436f 1cf88e5a39b3a257ed f4a22be3c955ac49da 2e2107b67a1924419 563	
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory

Semantic unit	1.5.2.3 messageDigestOriginator		
Semantic components	None		
Definition	The agent that created the original message digest that is compared in a fixity check.		
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 who calculated the initial value of the message digest.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		DRS A0000978	
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
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., “NRS” referring to the archive itself) or a pointer to an agent description (e.g., “A0000987” taken here to be an <i>agentIdentifierValue</i>).		

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.3 size		
Semantic components	None		
Definition	The size in bytes of the file or bitstream stored in the repository.		
Rationale	Size is useful for ensuring the correct number of bytes from storage have 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		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		2038937	
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
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.		

Semantic unit	1.5.4 format		
Semantic components	1.5.4.1 formatDesignation 1.5.4.2 formatRegistry 1.5.4.3 formatNote		
Definition	Identification of the format of a file or bitstream where format is the organization of digital information according to preset specifications.		
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 into a format registry, should be sufficient to associate the object with more detailed format information.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Mandatory	Mandatory
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 file name extension, recommended practice is to independently identify the format 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	<p>A bitstream embedded within a file may have different characteristics than the larger file. For example, a bitstream in LaTeX format could be embedded within an SGML file, or multiple images using different colorspace could be embedded within a TIFF file. <i>format</i> 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, <i>format</i> can be recorded for embedded bitstreams.</p> <p>Although this semantic unit is mandatory, both of its subunits are optional. At least one subunit (i.e. either <i>formatDesignation</i> or <i>formatRegistry</i>) must be present if this container is included or both may be used. If the subunit (<i>formatDesignation</i> or <i>formatRegistry</i>) needs to be repeated, the entire <i>format</i> container is repeated. This</p>		

THE PREMIS DATA DICTIONARY

	<p>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 <i>format</i> designations are recorded, each is given within a separate <i>format</i> container. The <i>format</i> container may also be repeated for multiple format registry entries.</p> <p>See “Format information,” page 195.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Semantic unit	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.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Usage notes	<p>Either <i>formatDesignation</i> or at least one instance of <i>formatRegistry</i> is required. Both may be included.</p> <p>The most specific format (or format profile) should be recorded. A repository (or formats registry) may wish to use multipart format names (e.g., “TIFF_GeoTIFF” or “WAVE_MPEG_BWF”) to achieve this specificity.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.4.1.1 formatName		
Semantic components	None		
Definition	A designation of the format of the file or bitstream.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		Text/sgml image/tiff/geotiff Adobe PDF DES PGP base64 unknown	LaTeX
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Usage notes	For unidentified formats, <i>formatName</i> may be recorded as “unknown”.		

Semantic unit	1.5.4.1.2 formatVersion		
Semantic components	None		
Definition	The version of the format named in <i>formatName</i> .		
Rationale	Many authority lists of format names are not granular enough to indicate version, for example, MIME Media types.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		6.0 2003	
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Usage notes	If the format is versioned, <i>formatVersion</i> should be recorded. It can be either a numeric or chronological designation.		

THE PREMIS DATA DICTIONARY

Semantic unit	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.		
Rationale	If central format registries are available to the preservation repository, they may provide an excellent way of referencing detailed format information.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Usage notes	<p>Either <i>formatDesignation</i> or at least one instance of <i>formatRegistry</i> is required. If more than one <i>formatRegistry</i> needs to be recorded the format container should be repeated to include each additional set of <i>formatRegistry</i> information.</p> <p>The PREMIS working group assumed that a number of format registries will be developed and maintained to support digital preservation efforts. The proposal for a Global Digital Format Registry (GDFR) (http://hul.harvard.edu/gdfr/documents.html#data), for example, would create a network-accessible registry designed to store detailed specifications on formats and profiles.</p>		

Semantic unit	1.5.4.2.1 formatRegistryName		
Semantic components	None		
Definition	A designation identifying the referenced format registry.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		PRONOM www.nationalarchives.gov.uk/pronom Representation Information Registry Repository	FRED: A format registry demonstration, release 0.07
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Usage notes	This can be a formal name, internally used name, or URI.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.4.2.2 formatRegistryKey		
Semantic components	None		
Definition	The unique key used to reference an entry for this format in a format registry.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		info:gdfn/fred/f/tiff TIFF/6.0	
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory

Semantic unit	1.5.4.2.3 formatRegistryRole		
Semantic components	None		
Definition	The purpose or expected use of the registry.		
Rationale	The same <i>format</i> may be defined in different registries for different purposes. For example, one registry may give detailed format specifications while another has profile information. 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.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		Specification Validation profile	
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.4.3 formatNote		
Semantic components	None		
Definition	Additional information about format.		
Rationale	Qualifying information may be needed to supplement format designation and registry information or record a status for identification		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		tentative identification disjunction multiple format identifications found	
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Usage notes	The <i>formatNote</i> may contain free text, a reference pointer, or a value from a controlled list.		

Semantic unit	1.5.5 creatingApplication		
Semantic components	1.5.5.1 creatingApplicationName 1.5.5.2 creatingApplicationVersion 1.5.5.3 dateCreatedByApplication 1.5.5.4 creatingApplicationExtension		
Definition	Information about the application that created the object.		
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 which rendering 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		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Creation / Maintenance notes	<p>If the object was created by the repository, assignment of creating application information should be straightforward.</p> <p>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.</p>		
Usage notes	<p>This semantic unit applies to both objects created external to the repository and subsequently ingested, and to objects created by the repository, for example, through migration events.</p> <p>The <i>creatingApplication</i> 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 file should be completely described as an Object entity and linked by using relationship information with a <i>relationshipType</i> “derivation.”</p> <p>It may also be repeated to record the creating application before the</p>		

THE PREMIS DATA DICTIONARY

	<p>object was ingested as well as the creating application used as part of the ingest process. For example, an HTML file was created pre-ingest using Dreamweaver, and the Web crawler Heritrix then captured a snapshot of the files as part of the ingest.</p> <p>The amount of information needed for <i>creatingApplication</i> given here is minimal. For more granularity, extensibility is provided.</p> <p>Rather than having each repository record this locally, it would be preferable to have a registry of this information similar to format or environment registries.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Semantic unit	1.5.5.1 creatingApplicationName		
Semantic components	None		
Definition	A designation for the name of the software program that created the object.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		MSWord	
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Usage notes	The <i>creatingApplication</i> 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 <i>creatingApplication</i> is Word, not the Ingest program.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.5.2 creatingApplicationVersion		
Semantic components	None		
Definition	The version of the software program that created the object.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		2000	1.4
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional

Semantic unit	1.5.5.3 dateCreatedByApplication		
Semantic components	None		
Definition	The actual or approximate date and time the object was created.		
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.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		2000-12-01 20030223T151047	
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Usage notes	<p>Use the most precise date available.</p> <p>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 <i>dateCreatedByApplication</i> of all three files is 2001. The date a file is written to storage can be recorded as an Event.</p> <p>If the object itself contains internal creation and modification dates, the modification date should be used as <i>dateCreatedByApplication</i>.</p> <p>If the application is a Web harvester capturing an object at a point of time, use for date captured.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.5.4 creatingApplicationExtension		
Semantic components	Defined externally		
Definition	Creating application information using semantic units defined external to PREMIS.		
Rationale	There may be a need to supplement or replace PREMIS defined units.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Usage notes	<p>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 externally defined schema, a reference to that schema must be provided. See further guidance “Extensibility,” page 19.</p> <p>If <i>creatingApplicationExtension</i> container needs to be associated explicitly with any PREMIS subunit under <i>creatingApplication</i>, the container <i>creatingApplication</i> is repeated. If extensions from different external schemas are needed, <i>creatingApplication</i> should also be repeated.</p>		

Semantic unit	1.5.6 inhibitors		
Semantic components	1.5.6.1 inhibitorType 1.5.6.2 inhibitorTarget 1.5.6.3 inhibitorKey		
Definition	Features of the object intended to inhibit access, use, or migration.		
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		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
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 often not possible to tell that a file has been encrypted by parsing it; the file may appear to be ASCII text. Therefore, information about inhibitors should be supplied as metadata with submitted objects when possible.		
Usage notes	<p>Some file formats allow encryption for embedded bitstreams.</p> <p>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.</p> <p>For certain types of inhibitor keys, more granularity may be required. If the inhibitor key information is identical to key information in digital signatures, use those semantic units.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.6.1 inhibitorType		
Semantic components	None		
Definition	The inhibitor method employed.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		DES PGP Blowfish Password protection	
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
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”.		

Semantic unit	1.5.6.2 inhibitorTarget		
Semantic components	None		
Definition	The content or function protected by the inhibitor.		
Data constraint	Values should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		All content Function: Play Function: Print	
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Usage notes	If not supplied, assume that the target is the content of the object.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.5.6.3 inhibitorKey		
Semantic components	None		
Definition	The decryption key or password.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		[DES decryption key]	
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Usage notes	The key should be provided if known. However, it is not advisable to actually store the <i>inhibitorKey</i> in plain text in an unsecure database.		

Semantic unit	1.5.7 objectCharacteristicsExtension		
Semantic components	Defined externally		
Definition	A container to include semantic units defined outside of PREMIS.		
Rationale	There may be a need to replace or extend PREMIS defined units.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Usage notes	<p>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,” page 19.</p> <p><i>objectCharacteristicsExtension</i> 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.</p> <p>If <i>objectCharacteristicsExtension</i> container needs to be associated explicitly with any PREMIS subunit under <i>objectCharacteristics</i>, the container <i>objectCharacteristics</i> is repeated. If extensions from different external schemas are needed, <i>objectCharacteristics</i> should also be repeated.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	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.		
Rationale	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		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Not applicable
Examples		N419.pdf	
Repeatability	Not repeatable	Not repeatable	
Obligation	Optional	Optional	
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 the name of the object as designated in the Submission Information Package (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 representation at the originating repository. In the case of representations, this may be a directory name.		

Semantic unit	1.7 storage		
Semantic components	1.7.1 <i>contentLocation</i> 1.7.2 <i>storageMedium</i>		
Definition	Information about how and where a file is stored in the storage system.		
Rationale	It is necessary for a repository to associate the <i>contentLocation</i> with the <i>storageMedium</i> .		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Mandatory	Mandatory
Usage notes	<p>Normally there would be a single storage location and medium for an object, because an object in another location would be considered a different object. The storage composite 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. They must have a single <i>objectIdentifier</i> and be managed as a single object by the repository.</p> <p>Although this semantic unit is mandatory, both of its subunits are optional. At least one subunit (i.e. either <i>contentLocation</i> or <i>storageMedium</i>) must be present or both may be used.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.7.1 contentLocation		
Semantic components	1.7.1.1 contentLocationType 1.7.1.2 contentLocationValue		
Definition	Information needed to retrieve a file from the storage system, or to access a bitstream within a file.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Creation / Maintenance notes	A preservation repository should never refer to content that it does not control. Therefore, the PREMIS working group assumed that the repository will always assign the <i>contentLocation</i> , probably by program.		
Usage notes	If the preservation repository uses the <i>objectIdentifier</i> as a handle for retrieving data, <i>contentLocation</i> is implicit and does not need to be recorded.		

Semantic unit	1.7.1.1 contentLocationType		
Semantic components	None		
Definition	The means of referencing the location of the content.		
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.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		URI hdl NTFS EXT3	byte offset
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	1.7.1.2 contentLocationValue		
Semantic components	None		
Definition	The reference to the location of the content used by the storage system.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		http://wwasearch.loc.gov/107th/200212107035/http://house.gov/langevin/ hdl:loc.pnp/cph.3b34188 c:\apache2\htdocs\index.html /home/web/public_html/index.html	64 [offset from start of file c:\apache2\htdocs\image\logo.gif]
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
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. It is up to the repository to determine the level of granularity that should be recorded.		

Semantic unit	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).		
Rationale	The repository needs to know the medium on which an object is stored in order to know how and when to do media refreshment and media migration.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		Magnetic tape Hard disk TSM	Magnetic tape Hard disk TSM
Repeatability		Not Repeatable	Not Repeatable
Obligation		Optional	Optional
Usage notes	<p>In some cases this can be masked from direct repository management by storage management systems but the underlying assumption is that the repository ultimately is in control and needs to manage for technological obsolescence.</p> <p>In some cases the value may not be the specific medium, but the system that knows the medium, e.g., Tivoli Storage Manager (TSM).</p> <p>Knowing the storage medium is an internal requirement in order to trigger preservation actions. However, since this is not information that is used for exchange purposes, it is optional.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8 environment		
Semantic components	1.8.1 environmentCharacteristic 1.8.2 environmentPurpose 1.8.3 environmentNote 1.8.4 dependency 1.8.5 software 1.8.6 hardware 1.8.7 environmentExtension		
Definition	Hardware/software combinations supporting use of the object.		
Rationale	Environment is the means by which the user renders and interacts with content. Separation of digital content from its environmental context can result in the content becoming unusable.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Creation / Maintenance notes	<p>This information may be omitted when the repository is doing only bit-level preservation on the object.</p> <p>Rather than having each repository record this locally, it would be preferable to have a registry of environment information similar to proposed registries of format information.</p> <p>Repositories may choose to design mechanisms for inheritance, so that if the environment required for each file within a representation is identical to the environment recorded for the representation as a whole, it is not necessary to store this information in each file.</p> <p>See “Environment,” page 197.</p>		
Usage notes	All of this semantic units’ subunits are optional. At least one subunit (i.e. <i>environmentNote</i> , <i>dependency</i> , <i>software</i> , <i>hardware</i> , and/or <i>environmentExtension</i>) must be present if this container is included.		

Semantic unit	1.8.1 environmentCharacteristic		
Semantic components	None		
Definition	An assessment of the extent to which the described environment supports its purpose.		
Rationale	If multiple environments are described, this element can help to distinguish among them.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	unspecified minimum	recommended minimum	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
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 environments registry, <i>environmentCharacteristic</i> might also be obtained from the registry. Note however that the criteria for “recommended” may be different for different repositories.		
Usage notes	<p>Suggested values:</p> <p>unspecified = no attempt made to provide this value</p> <p>known to work = the object can be rendered in this environment</p> <p>minimum = the least demanding (in terms of components or resources needed) environment known to work by the repository</p> <p>recommended = an environment preferred for optional rendering</p> <p>If an environment is both “minimum” and “recommended,” use “recommended.”</p> <p>“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.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.2 environmentPurpose		
Semantic components	None		
Definition	The use(s) supported by the specified environment.		
Rationale	Different environments can support different uses of objects. For example, the environment needed to edit and modify a file can be quite different than the environment needed to render it.		
Data constraint	Values should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Creation / Maintenance notes	This value would have to be supplied by the agent that provided the hardware and software environment information, which might be the submitter, the repository, or an environments registry.		
Usage notes	<p>Suggested values: render, edit.</p> <p>This list may need to be expanded. Other values might indicate the ability to transform, print, and manipulate by program.</p>		

Semantic unit	1.8.3 environmentNote		
Semantic components	None		
Definition	Additional information about the environment.		
Rationale	There may be a need to give a textual description of the environment for additional explanation.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		This environment assumes that the PDF will be stored locally and used with a standalone PDF reader.	
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	<p>This note could be used to record the context of the environment information. For example, if a file can be rendered through a PC client application or through a browser with a plug-in, this note could be used to identify which situation applies.</p> <p>The note should not be used for a textual description of environment information recorded more rigorously elsewhere.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.4 dependency		
Semantic components	1.8.4.1 dependencyName 1.8.4.2 dependencyIdentifier		
Definition	Information about a non-software component or associated file needed in order to use or render the representation or file, for example, a schema, a DTD, or an entity file declaration.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Creation / Maintenance notes	Recommended practice is for a repository to archive objects on which other objects depend. These may be sent by the submitter of the primary object, or they may in some cases be automatically obtained by the repository. For example, a markup file will often contain links to other objects it requires such as DTDs or XML Schema. If it does, these objects can often be identified by the link and downloaded by the repository.		
Usage notes	<p>This semantic unit is for additional objects that are necessary to render a file or representation, not for required software or hardware. It may also be used for a non-executable component of the object, such as a font or style sheet. For things that the software requires, see <i>swDependency</i>, page 94.</p> <p>This semantic unit does not include objects required by structural relationships, such as child content objects (e.g., figures that are part of an article), which are recorded under relationship with a <i>relationshipType</i> of “structural”.</p> <p>It is up to the repository to determine what constitutes a dependency in the context of the designated community.</p> <p>The objects noted may be internal or external to the preservation repository.</p>		

Semantic unit	1.8.4.1 dependencyName		
Semantic components	None		
Definition	A designation for a component or associated file needed by the representation or file.		
Rationale	It may not be self-evident from the <i>dependencyIdentifier</i> what the name of the object actually is.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		Additional Element Set for Language Corpora	
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.4.2 dependencyIdentifier		
Semantic components	1.8.4.2.1 dependencyIdentifierType 1.8.4.2.2 dependencyIdentifierValue		
Definition	A unique designation used to identify a dependent resource.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatability	Repeatability	Repeatability
Obligation	Optional	Optional	Optional
Usage notes	The <i>dependencyIdentifier</i> must be unique within the preservation repository, although it might not be globally unique.		

Semantic unit	1.8.4.2.1 dependencyIdentifierType		
Semantic components	None		
Definition	A designation of the domain in which the identifier of the dependent resource is unique.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		URI	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	A preservation repository needs to know both the type of object identifier and the value. When 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 explicitly.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.4.2.2 dependencyIdentifierValue		
Semantic components	None		
Definition	The value of the <i>dependencyIdentifier</i> .		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		http://www.tei-c.org/P4X/DTD/teicorp2.dtd	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

Semantic unit	1.8.5 software		
Semantic components	1.8.5.1 swName 1.8.5.2 swVersion 1.8.5.3 swType 1.8.5.4 swOtherInformation 1.8.5.5 swDependency		
Definition	Software required to render or use the object.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Creation / Maintenance notes	<p>If recording this explicitly, many different software environments may apply; for example, a particular object such as a PDF file may be viewable by several versions of several applications running under several operating systems and operating system versions. Although at least one software environment should be recorded, it is not necessary to record them all and each repository will have to make its own decisions about which software environments to record.</p> <p>Also, what appears to the user as a single rendering program can have many dependencies, including system utilities, runtime libraries, and so on, which each might have their own dependencies in turn.</p> <p>As with environment, metadata may be more efficiently managed in conjunction with a format registry either internal or external to a repository. In the absence of a global mechanism, repositories may be forced to develop their own local “registries” relating format to software environment.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.5.1 swName		
Semantic components	None		
Definition	Manufacturer and title of the software application.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	Sybase	Adobe Photoshop Adobe Acrobat Reader	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	Include manufacturer when this helps to identify or disambiguate the product, for example, use “Adobe Photoshop” rather than “Photoshop.”		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.5.2 swVersion		
Semantic components	None		
Definition	The version or versions of the software referenced in <i>swName</i> .		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		>=2.2.0 6.0 2000	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Usage notes	If there is no formal version, the date of issuance can be used.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.5.3 swType		
Semantic components	None		
Definition	Class or category of <i>software</i> .		
Rationale	Several different layers of software can be required to support an object.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	<p>Suggested values:</p> <p>renderer = application that can display/play/execute the format instance, e.g., image viewer, video player, Java virtual machine (when the format instance is a Java class file)</p> <p>ancillary = required ancillary software, e.g., run time libraries, browser plug-ins, compression/decompression routines, utilities, operating system emulators, etc.</p> <p>operatingSystem = software that supports application execution, process scheduling, memory management, file systems, etc.</p> <p>driver = software with the primary function of communicating between hardware and the operating system or other software</p>		

Semantic unit	1.8.5.4 swOtherInformation		
Semantic components	None		
Definition	Additional requirements or instructions related to the software referenced in <i>swName</i> .		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		Install Acroread (Adobe Acrobat) first; copy nppdf.so (the plug-in) to your Mozilla plug-ins directory, and make sure a copy of (or symlink to) Acroread is in your PATH.	
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	This could be a reliable persistent identifier or URI pointing to software documentation within or outside the repository.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.5.5 swDependency		
Semantic components	None		
Definition	The name and, if applicable, version of any software component needed by the software referenced in <i>swName</i> in the context of using this object.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		GNU gcc >= 2.7.2	
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	The value should be constructed in a way that is consistent with the construction of <i>swName</i> and <i>swVersion</i> . This semantic unit identifies the software that is needed by what is recorded in <i>swName</i> , for example, a Perl script that depends on a Perl module. In this case the Perl script is listed in <i>swName</i> , with the module in <i>swDependency</i> within a software container.		

Semantic unit	1.8.6 hardware		
Semantic components	1.8.6.1 hwName 1.8.6.2 hwType 1.8.6.3 hwOtherInformation		
Definition	Hardware components needed by the software referenced in <i>swName</i> or the human user of the referenced software.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Creation / Maintenance notes	<p>Hardware environment information can be very difficult to provide. Many different hardware environments may apply; there are a huge number of combinations of maker and type of CPU, memory, video drivers, and so on. Although at least one hardware environment should be recorded, it is not necessary to record them all and each repository will have to make its own decisions about which hardware environments to record.</p> <p>Because of the difficulty recording this information comprehensively, it would be optimal if central registries of environment information existed. In many cases the environment of a file object is directly associated with the format, making registry lookup by format feasible. In the absence of a global mechanism, repositories may be forced to develop their own local “registries” relating format to <i>hwEnvironment</i>.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.6.1 hwName		
Semantic components	None		
Definition	Manufacturer, model, and version (if applicable) of the hardware.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		Intel Pentium III 1 GB DRAM Windows XP-compatible joystick	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	<p>Include manufacturer when this helps to identify or disambiguate the product.</p> <p>Include version for firmware or other components where that information is pertinent.</p>		

Semantic unit	1.8.6.2 hwType		
Semantic components	None		
Definition	Class or category of the hardware.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	Suggested values: processor, memory, input/output device, storage device.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.8.6.3 hwOtherInformation		
Semantic components	None		
Definition	Additional requirements or instructions related to the hardware referenced in <i>hwName</i> .		
Rationale	For hardware, the amount of computing resource needed (such as memory, storage, processor speed, etc.) may need to be documented. In addition, more detailed instructions may be needed to install and/or operate the hardware.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	32MB minimum	32MB minimum Required RAM for Apache is unknown	
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	This could be an identifier or URI used to point to hardware documentation.		

Semantic unit	1.8.7 environmentExtension		
Semantic components	Defined externally		
Definition	A container to include semantic units defined outside of PREMIS.		
Rationale	There may be a need to replace or extend PREMIS defined units.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples			
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	<p>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,” page 19.</p> <p>If <i>environmentExtension</i> container needs to be associated explicitly with any PREMIS subunit under <i>environment</i>, the container <i>environment</i> is repeated. If extensions from different external schemas are needed, <i>environment</i> should also be repeated.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.9 signatureInformation		
Semantic components	1.9.1 signature 1.9.2 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.		
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		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Usage notes	Either <i>signature</i> or <i>signatureInformationExtension</i> may be used. Use of <i>signatureInformationExtension</i> with the schema defined in W3C's <i>XML-Signature Syntax and Processing</i> (www.w3.org/TR/2002/REC-xmlsig-core-20020212/) is encouraged when applicable. See the discussion of digital signatures on page 201 for more information on use of both PREMIS-defined and externally-defined semantic units.		

Semantic unit	1.9.1 signature		
Semantic components	1.9.1.1 signatureEncoding 1.9.1.2 signer 1.9.1.3 signatureMethod 1.9.1.4 signatureValue 1.9.1.5 signatureValidationRules 1.9.1.6 signatureProperties 1.9.1.7 keyInformation		
Definition	Information needed to use a digital signature to authenticate the signer of an object and/or the information contained in the object.		
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		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Usage notes	Several of the semantic components of <i>signatureInformation</i> are taken from the W3C's <i>XML-Signature Syntax and Processing</i> ; see www.w3.org/TR/2002/REC-xmlsig-core-20020212/ for more information on the structure and application of these semantic units. (See also the discussion of digital signatures, page 201.)		

THE PREMIS DATA DICTIONARY

Semantic unit	1.9.1.1 signatureEncoding		
Semantic components	None		
Definition	The encoding used for the values of <i>signatureValue</i> , <i>keyInformation</i> .		
Rationale	These values cannot be interpreted correctly if the encoding is unknown.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		Base64 Ds:CryptoBinary	
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory

Semantic unit	1.9.1.2 signer		
Semantic components	None		
Definition	The individual, institution, or authority responsible for generating the signature.		
Rationale	The signer might also be carried in the <i>keyInformation</i> , but it can be accessed more conveniently if recorded here.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Usage notes	If the signer is an <i>Agent</i> known to the repository, an <i>agentIdentifier</i> can be used here.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.9.1.3 signatureMethod		
Semantic components	None		
Definition	A designation for the encryption and hash algorithms used for signature generation.		
Rationale	The same algorithms must be used for signature validation.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		DSA-SHA1 RSA-SHA1	
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Usage notes	Recommended practice is to encode the encryption algorithm first, followed by a hyphen, followed by the hash (message digest) algorithm.		

Semantic unit	1.9.1.4 signatureValue		
Semantic components	None		
Definition	The digital signature; a value generated from the application of a private key to a message digest.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Examples		juS5RhJ884qoFR 8flVXd/rbrSDVGn 40CapgB7qeQiT +rr0NekEQ6BHh UA8dT3+BCTBU QI0dBjlm19lwzEN XvS83zRECjzXb MRTUtVZiPZG2p qKPnL2YU3A964 5UCjTXU+jgFum v7k78hieAGDzNc i+PQ9KRmm//icT 7JaYztgt4=	
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	1.9.1.5 signatureValidationRules		
Semantic components	None		
Definition	The operations to be performed in order to validate the digital signature.		
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		
Object category	Representation	File	Bitstream
Applicability		Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Mandatory	Mandatory
Usage notes	<p>This may include the canonicalization method used before calculating the message digest, if the object was normalized before signing.</p> <p>This value could also be a pointer to archive documentation.</p>		

Semantic unit	1.9.1.6 signatureProperties		
Semantic components	None		
Definition	Additional information about the generation of the signature.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
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 <i>signatureProperties</i> .		

THE PREMIS DATA DICTIONARY

Semantic unit	1.9.1.7 keyInformation		
Semantic components	Extensible container		
Definition	Information about the signer's public key needed to validate the digital signature.		
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 (<i>signatureValue</i>) is correct. The repository must therefore have the public key value and some assurance that it truly belongs to the signer.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Not repeatable	Not repeatable
Obligation		Optional	Optional
Usage notes	<p>Different types of keys will have different structures and parameters. PREMIS does not define structure for this container. Recommended practice is to represent key values as defined for "KeyInfo" in the W3C's <i>XML-Signature Syntax and Processing</i> (www.w3.org/TR/2002/REC-xmlsig-core-20020212/).</p>		

Semantic unit	1.9.2 signatureInformationExtension		
Semantic components	Defined externally		
Definition	Digital signature information using semantic units defined outside of PREMIS.		
Rationale	There may be a need to replace or extend PREMIS defined units.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability		Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Optional	Optional
Usage notes	<p>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,” page 19.</p> <p>If <i>signatureInformationExtension</i> container needs to be associated explicitly with any PREMIS subunit under <i>signatureInformation</i>, the container <i>signatureInformation</i> is repeated. If extensions from different external schemas are needed, <i>signatureInformation</i> should also be repeated.</p> <p>Use of the W3C’s <i>XML-Signature Syntax and Processing</i> (www.w3.org/TR/2002/REC-xmlsig-core-20020212/) is encouraged when applicable.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.10 relationship		
Semantic components	1.10.1 relationshipType 1.10.2 relationshipSubType 1.10.3 relatedObjectIdentification 1.10.4 relatedEventIdentification		
Definition	Information about a relationship between this object and one or more other objects.		
Rationale	A preservation repository must know how to assemble complex objects from component parts (structural relationships) and rigorously track digital provenance (derivation relationships). Documentation about relationships between different objects is crucial to these purposes.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	<p>Most preservation repositories will want to record all relevant relationships.</p> <p>In complex scenarios, PREMIS might not be able to express rich enough structural relationships to be the only source of structural metadata.</p> <p>Many formats for representing structural information may be used instead of the semantic units specified here. This information must be known, and some implementations may know it by using other structures.</p> <p>Structural relationships at the file level are necessary to reconstruct a representation in order to ascertain that the representation is renderable.</p> <p>A record of structural relationships at the representation level may be necessary to render the representation.</p> <p>Structural relationships at the bitstream level can relate bitstreams within a file.</p> <p>Derivative relationships at the file and representation level are important for documenting digital provenance.</p>		

Semantic unit	1.10.1 relationshipType		
Semantic components	None		
Definition	A high-level categorization of the nature of the relationship.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	<p>Suggested values:</p> <p>structural = a relationship between parts of an object</p> <p>derivation = a relationship where one object is the result of a transformation performed on the related object</p> <p>A repository may find it necessary to define additional relationship types.</p>		

THE PREMIS DATA DICTIONARY

Semantic unit	1.10.2 relationshipSubType		
Semantic components	None		
Definition	A specific characterization of the nature of the relationship documented in <i>relationshipType</i> .		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	<p>Suggested values:</p> <p>has sibling = the object shares a common parent with the related object</p> <p>is part of = the object is contained by the related object (when these are the same entity types)</p> <p>has part = the object contains the related object (when these are the same entity types)</p> <p>is source of = the related object is a version of this object created by a transformation</p> <p>has source = the object is derived from the related object as a result of a transformation</p> <p>has root = for a representation only, the related object is the file that must be processed first in order to render the representation</p> <p>includes = for the relationship of a representation to a file, , or a file to a bitstream, the described object includes the referenced object</p> <p>is included in = for the relationship of a file to a representation, or a bitstream to a file, the described object is included in the referenced object</p> <p>A repository may find it necessary to define more or less granular relationships. For derivation relationships, note that the precise relationship may be indicated by the type of the related event.</p> <p>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.</p>		

Semantic unit	1.10.3 relatedObjectIdentification		
Semantic components	1.10.3.1 relatedObjectIdentifierType 1.10.3.2 relatedObjectIdentifierValue 1.10.3.3 relatedObjectSequence		
Definition	The identifier and sequential context of the related resource.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Mandatory	Mandatory	Mandatory
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 object outside. Internal and external references should be clear.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.10.3.1 relatedObjectIdentifierType		
Semantic components	None		
Definition	A designation of the domain within which the identifier is unique.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	[see examples for <i>objectIdentifierType</i>]	[see examples for <i>objectIdentifierType</i>]	[see examples for <i>objectIdentifierType</i>]
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	If the related object is held within the preservation repository, this should be the value of that object's <i>objectIdentifierType</i> .		

Semantic unit	1.10.3.2 relatedObjectIdentifierValue		
Semantic components	None		
Definition	The value of the related object identifier.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	[see examples for <i>objectIdentifierValue</i>]	[see examples for <i>objectIdentifierValue</i>]	[see examples for <i>objectIdentifierValue</i>]
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	If the related object is held within the preservation repository, this should be the value of that object's <i>objectIdentifierValue</i> .		

THE PREMIS DATA DICTIONARY

Semantic unit	1.10.3.3 relatedObjectSequence		
Semantic components	None		
Definition	The order of the related object relative to other objects with the same type of relationship.		
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 a page-image book, it is necessary to know the order of files representing pages.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		1 2 3	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Usage notes	<p>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 <i>relationshipSubType</i> might imply the sequence (e.g., “is preceding sibling,” “is following sibling”).</p> <p>There is no requirement that sequence numbers must be unique or sequential.</p> <p>Some related objects have no inherent sequence, for example, unordered Web pages making up a Web site. In this case all related objects can be given the “dummy” sequence number zero.</p> <p>This semantic unit is applicable only for structural relationships and is thus optional.</p>		

Semantic unit	1.10.4 relatedEventIdentification		
Semantic components	1.10.4.1 relatedEventIdentifierType 1.10.4.2 relatedEventIdentifierValue 1.10.4.3 relatedEventSequence		
Definition	The identifier and contextual sequence of an event associated with the relationship.		
Rationale	An object may be related to another object because of an event, for example, migration.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	For derivative relationships between objects <i>relatedEventIdentification</i> must be recorded.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.10.4.1 relatedEventIdentifierType		
Semantic components	None		
Definition	The <i>eventIdentifierType</i> of the related event.		
Data constraint	Must be an existing <i>eventIdentifierType</i> value.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	[see examples for <i>eventIdentifierType</i>]	[see examples for <i>eventIdentifierType</i>]	[see examples for <i>eventIdentifierType</i>]
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	For most preservation repositories, the <i>eventIdentifierType</i> will simply be its own internal numbering system. It can be implicit within the system and provided explicitly only if the data is exported.		

Semantic unit	1.10.4.2 relatedEventIdentifierValue		
Semantic components	None		
Definition	The <i>eventIdentifierValue</i> of the related event.		
Data constraint	Must be an existing <i>eventIdentifierValue</i> value.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	[see examples for <i>eventIdentifierValue</i>]	[see examples for <i>eventIdentifierValue</i>]	[see examples for <i>eventIdentifierValue</i>]
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	1.10.4.3 relatedEventSequence		
Semantic components	None		
Definition	The order of the related event.		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		1 2 3	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Optional	Optional	Optional
Usage notes	The sequence of a related event can be inferred from the <i>eventDateTime</i> associated with the related event.		

Semantic unit	1.11 linkingEventIdentifier		
Semantic components	1.11.1 linkingEventIdentifierType 1.11.2 linkingEventIdentifierValue		
Definition	The <i>eventIdentifier</i> of an event associated with the object.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	Use to link to events that are not associated with relationships between objects, such as format validation, virus checking, etc.		

THE PREMIS DATA DICTIONARY

Semantic unit	1.11.1 linkingEventIdentifierType		
Semantic components	None		
Definition	The <i>eventIdentifierType</i> value of the related event.		
Data constraint	Must be an existing <i>eventIdentifierType</i> value.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	[see examples for <i>eventIdentifierType</i>]	[see examples for <i>eventIdentifierType</i>]	[see examples for <i>eventIdentifierType</i>]
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory
Usage notes	For most preservation repositories, the <i>eventIdentifierType</i> will simply be their own internal numbering system. It can be implicit within the system and provided explicitly only if the data is exported.		

Semantic unit	1.11.2 linkingEventIdentifierValue		
Semantic components	None		
Definition	The <i>eventIdentifierValue</i> value of the related event.		
Data constraint	Must be an existing <i>eventIdentifierValue</i> value.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	[see examples for <i>eventIdentifierValue</i>]	[see examples for <i>eventIdentifierValue</i>]	[see examples for <i>eventIdentifierValue</i>]
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	1.12 linkingIntellectualEntityIdentifier		
Semantic components	1.12.1 linkingIntellectualEntityIdentifierType 1.12.2 linkingIntellectualEntityIdentifierValue		
Definition	An identifier for an intellectual entity associated with the object.		
Data constraint	Container		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional
Usage notes	Use to link to an intellectual entity that is related to the object. This may be a link to descriptive metadata that describes the intellectual entity or some other surrogate for it that can be referenced. This link will likely be to an identifier of an object that is at a higher conceptual level than the object for which the metadata is provided, for example, to a collection or parent object.		

Semantic unit	1.12.1 linkingIntellectualEntityIdentifierType		
Semantic components	None		
Definition	A designation of the domain within which the <i>linkingIntellectualEntityIdentifier</i> is unique.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		URI LCCN	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	1.12.2 linkingIntellectualEntityIdentifierValue		
Semantic components	None		
Definition	The value of the <i>linkingIntellectualEntityIdentifier</i> .		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples	hdl:loc.natlib/mrva00 02.0495 info:lccn/19018302		
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

Semantic unit	1.13 linkingRightsStatementIdentifier		
Semantic components	1.13.1 linkingRightsStatementIdentifierType 1.13.2 linkingRightsStatementIdentifierValue		
Definition	An identifier for a rights statement associated with the object.		
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		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Repeatable	Repeatable	Repeatable
Obligation	Optional	Optional	Optional

THE PREMIS DATA DICTIONARY

Semantic unit	1.13.1 linkingRightsStatementIdentifierType		
Semantic components	None		
Definition	A designation of the domain within which the <i>linkingRightsStatementIdentifier</i> is unique.		
Data constraint	Value should be taken from a controlled vocabulary.		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Examples		URI LCCN	
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

Semantic unit	1.13.2 linkingRightsStatementIdentifierValue		
Semantic components	None		
Definition	The value of the <i>linkingRightsStatementIdentifier</i> .		
Data constraint	None		
Object category	Representation	File	Bitstream
Applicability	Applicable	Applicable	Applicable
Repeatability	Not repeatable	Not repeatable	Not repeatable
Obligation	Mandatory	Mandatory	Mandatory

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 eventDetail (O, NR)
- 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)

Semantic unit	2.1 eventIdentifier
Semantic components	2.1.1 eventIdentifierType 2.1.2 eventIdentifierValue
Definition	A designation used to uniquely identify the event within the preservation repository system.
Rationale	Each event recorded by the preservation archive must have a unique identifier to allow it to be related to objects, agents, and other events.
Data constraint	Container
Repeatability	Not repeatable
Obligation	Mandatory
Creation / Maintenance notes	The <i>eventIdentifier</i> is likely to be system generated. There is no global scheme or standard for these identifiers. The identifier is therefore not repeatable.

THE PREMIS DATA DICTIONARY

Semantic unit	2.1.1 eventIdentifierType
Semantic components	None
Definition	A designation of the domain within which the event identifier is unique.
Data constraint	None
Examples	FDA Stanford Repository Event ID UUID
Repeatability	Not repeatable
Obligation	Mandatory
Creation / Maintenance notes	For most preservation repositories, the <i>eventIdentifierType</i> will be its own internal numbering system. It can be implicit within the system and provided explicitly only if the data is exported.

Semantic unit	2.1.2 eventIdentifierValue
Semantic components	None
Definition	The value of the <i>eventIdentifier</i> .
Data constraint	None
Examples	[a binary integer] E-2004-11-13-000119 58f202ac-22cf-11d1-b12d-002035b29092
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	2.2 eventType
Semantic components	None
Definition	A categorization of the nature of the event.
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.
Examples	E77 [a code used within a repository for a particular event type] Ingest
Repeatability	Not repeatable
Obligation	Mandatory
Usage notes	<p>Each repository should define its own controlled vocabulary of <i>eventType</i> values. A suggested starter list for consideration (see also the Glossary for more detailed definitions):</p> <p>capture = the process whereby a repository actively obtains an object</p> <p>compression = the process of coding data to save storage space or transmission time</p> <p>creation = the act of creating a new object</p> <p>deaccession = the process of removing an object from the inventory of a repository</p> <p>decompression = the process of reversing the effects of compression</p> <p>decryption = the process of converting encrypted data to plaintext</p> <p>deletion = the process of removing an object from repository storage</p> <p>digital signature validation = the process of determining that a decrypted digital signature matches an expected value</p> <p>dissemination = the process of retrieving an object from repository storage and making it available to users</p> <p>fixity check = the process of verifying that an object has not been changed in a given period</p> <p>ingestion = the process of adding objects to a preservation repository</p> <p>message digest calculation = the process by which a message digest ("hash") is created</p> <p>migration = a transformation of an object creating a version in a more contemporary format</p>

	<p>normalization = a transformation of an object creating a version more conducive to preservation</p> <p>replication = the process of creating a copy of an object that is, bit-wise, identical to the original</p> <p>validation = the process of comparing an object with a standard and noting compliance or exceptions</p> <p>virus check = the process of scanning a file for malicious programs</p> <p>Note that migration, normalization, and replication are more precise subtypes of the creation event. “Creation” can be used when more precise terms do not apply, for example, when a digital object was first created by scanning from paper.</p> <p>In general, the level of specificity in recording the type of event (e.g., whether the <i>eventType</i> indicates a transformation, a migration or a particular method of migration) is implementation specific and will depend upon how reporting and processing is done.</p> <p>Recommended practice is to record detailed information about the event itself in <i>eventDetail</i> rather than using a very granular value for <i>eventType</i>.</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

THE PREMIS DATA DICTIONARY

Semantic unit	2.3 eventDateTime
Semantic components	None
Definition	The single date and time, or date and time range, at or during which the event occurred.
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.
Examples	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]
Repeatability	Not repeatable
Obligation	Mandatory
Usage notes	Recommended practice is to record the most specific time possible and to designate the time zone.

Semantic unit	2.4 eventDetail
Semantic components	None
Definition	Additional information about the event.
Data constraint	None
Examples	Object permanently withdrawn by request of Caroline Hunt. Program="MIGJP2JP2K"; version="2.2"
Repeatability	Not repeatable
Obligation	Optional
Usage notes	<i>eventDetail</i> is not intended to be processed by machine. It may record any information about an event and/or point to information stored elsewhere.

THE PREMIS DATA DICTIONARY

Semantic unit	2.5 eventOutcomeInformation
Semantic components	2.5.1 eventOutcome 2.5.2 eventOutcomeDetail
Definition	Information about the outcome of an event.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional
Usage notes	<p>A repository may wish to supplement a coded <i>eventOutcome</i> value with additional information in <i>eventOutcomeDetail</i>. Since events may have more than one outcome, the container is repeatable.</p> <p>All subunits of this semantic unit are optional. At least one subunit (i.e. <i>eventOutcome</i> or <i>eventOutcomeDetail</i>) must be present if this container is included.</p>

Semantic unit	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.
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.
Examples	00 [a code meaning “action successfully completed”] CV-01 [a code meaning “checksum validated”]
Repeatability	Not repeatable
Obligation	Optional
Usage notes	<p>Recommended practice is to use a controlled vocabulary that a system can act upon automatically. More detail about the outcome may be recorded in <i>eventOutcomeDetail</i>.</p> <p>Recommended practice is to define events with sufficient granularity that each event has a single outcome.</p>

THE PREMIS DATA DICTIONARY

Semantic unit	2.5.2 eventOutcomeDetail
Semantic components	2.5.2.1 eventOutcomeDetailNote 2.5.2.2 eventOutcomeDetailExtension
Definition	A detailed description of the result or product of the event.
Rationale	An event outcome may be sufficiently complex that a coded description is not adequate to document it.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional
Usage notes	<p>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.</p> <p>If the event was a validity check (e.g., profile conformance) any anomalies or quirks discovered would be recorded here.</p> <p>All subunits of this semantic unit are optional. At least one subunit (i.e. <i>eventOutcomeDetailNote</i> and/or <i>eventOutcomeDetailExtension</i>) must be present if this container is included.</p>

Semantic unit	2.5.2.1 eventOutcomeDetailNote
Semantic components	None
Definition	A detailed description of the result or product of the event in textual form.
Rationale	Additional information in textual form may be needed about the outcome of the event.
Data constraint	None
Examples	LZW compressed file Non-standard tags found in header
Repeatability	Not repeatable
Obligation	Optional

THE PREMIS DATA DICTIONARY

Semantic unit	2.5.2.2 eventOutcomeDetailExtension
Semantic components	Defined externally
Definition	A container to include semantic units defined outside of PREMIS.
Rationale	There may be a need to replace or extend PREMIS defined units.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional
Usage notes	<p>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,” page 19.</p> <p>If <i>eventOutcomeDetailExtension</i> container needs to be associated explicitly with any PREMIS subunit under <i>eventOutcomeDetail</i>, the container <i>eventOutcomeDetail</i> is repeated. If extensions from different external schemas are needed, <i>eventOutcomeDetail</i> should also be repeated.</p>

Semantic unit	2.6 linkingAgentIdentifier
Semantic components	2.6.1 linkingAgentIdentifierType 2.6.2 linkingAgentIdentifierValue 2.6.3 linkingAgentRole
Definition	Information about an agent associated with an event.
Rationale	Digital provenance requires often that relationships between agents and events are documented.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional
Usage notes	Recommended practice is to record the agent if possible.

THE PREMIS DATA DICTIONARY

Semantic unit	2.6.1 linkingAgentIdentifierType
Semantic components	None
Definition	A designation of the domain in which the linking agent identifier is unique.
Data constraint	Value should be taken from a controlled vocabulary.
Examples	[see examples for agentIdentifierType]
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	2.6.2 linkingAgentIdentifierValue
Semantic components	None
Definition	The value of the linking agent identifier.
Data constraint	None
Examples	[see examples for <i>agentIdentifierValue</i>]
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	2.6.3 linkingAgentRole
Semantic components	None
Definition	The role of the agent in relation to this event.
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.
Examples	Authorizer Implementer Validator Executing program
Repeatability	Repeatable
Obligation	Optional

Semantic unit	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.
Rationale	Digital provenance often requires that relationships between objects and events are documented.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional

THE PREMIS DATA DICTIONARY

Semantic unit	2.7.1 linkingObjectIdentifierType
Semantic components	None
Definition	A designation of the domain in which the linking object identifier is unique.
Data constraint	Value should be taken from a controlled vocabulary.
Examples	[see examples for <i>objectIdentifierType</i>]
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	2.7.2 linkingObjectIdentifierValue
Semantic components	None
Definition	The value of the linking object identifier.
Data constraint	None
Examples	[see examples for <i>objectIdentifierValue</i>]
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	2.7.3 linkingObjectRole
Semantic components	None
Definition	The role of the object associated with an event.
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	None
Examples	source outcome
Repeatability	Repeatable
Obligation	Optional

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.

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 (R, M)
 - 3.1.1 agentIdentifierType (M, NR)
 - 3.1.2 agentIdentifierValue (M, NR)
- 3.2 agentName (O, R)
- 3.3 agentType (O, NR)

THE PREMIS DATA DICTIONARY

Semantic unit	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.
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
Repeatability	Repeatable
Obligation	Mandatory
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 <i>agentIdentifier</i> is repeatable in order to allow both repository-assigned and externally-assigned identifiers to be recorded. See Creation/Maintenance note above.

Semantic unit	3.1.1 agentIdentifierType
Semantic components	None
Definition	A designation of the domain in which the agent identifier is unique.
Data constraint	Value should be taken from a controlled vocabulary.
Examples	LCNAF SAN MARC Organization Codes URI
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	3.1.2 agentIdentifierValue
Semantic components	None
Definition	The value of the <i>agentIdentifier</i> .
Data constraint	None
Examples	92-79971 Owens, Erik C. 234-5676 MH-CS info:lccn/n78890351
Repeatability	Not repeatable
Obligation	Mandatory
Usage notes	May be a unique key or a controlled textual form of name.

Semantic unit	3.2 agentName
Semantic components	None
Definition	A text string which could be used in addition to <i>agentIdentifier</i> to identify an agent.
Rationale	This semantic unit provides a more reader-friendly version of the agent identified by the agentIdentifier.
Data constraint	None
Examples	Erik Owens Woodyard Pc
Repeatability	Repeatable
Obligation	Optional
Usage notes	The value is not necessarily unique.

THE PREMIS DATA DICTIONARY

Semantic unit	3.3 agentType
Semantic components	None
Definition	A high-level characterization of the type of agent.
Data constraint	Value should be taken from a controlled vocabulary.
Repeatability	Not repeatable
Obligation	Optional
Usage notes	Suggested values: person organization software

Rights Entity

For the purpose of the PREMIS Data Dictionary, statements of rights and permissions are taken to be constructs that can be described as the Rights entity. Rights are entitlements allowed to 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 rights statements 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 be granted by copyright law, by statute, or by a license agreement with the rightsholder.

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)
 - 4.1.4 *licenseInformation* (O, NR)
 - 4.1.4.1 *licenseIdentifier* (O, NR)
 - 4.1.4.1.1 *licenseIdentifierType* (M, NR)
 - 4.1.4.1.2 *licenseIdentifierValue* (M, NR)
 - 4.1.4.2 *licenseTerms* (O, NR)
 - 4.1.4.3 *licenseNote* (O, R)
 - 4.1.5 *statuteInformation* (O, R)
 - 4.1.5.1 *statuteJurisdiction* (M, NR)

THE PREMIS DATA DICTIONARY

- 4.1.5.2 statuteCitation (M, NR)
 - 4.1.5.3 statuteInformationDeterminationDate (O, NR)
 - 4.1.5.4 statuteNote (O, R)
 - 4.1.6 rightsGranted (O, R)
 - 4.1.6.1 act (M, NR)
 - 4.1.6.2 restriction (O, R)
 - 4.1.6.3 termOfGrant (M, NR)
 - 4.1.6.3.1 startDate (M, NR)
 - 4.1.6.3.2 endDate (O, NR)
 - 4.1.6.4 rightsGrantedNote (O, R)
 - 4.1.7 linkingObjectIdentifier (O, R)
 - 4.1.7.1 linkingObjectIdentifierType (M, NR)
 - 4.1.7.2 linkingObjectIdentifierValue (M, NR)
 - 4.1.8 linkingAgentIdentifier (O, R)
 - 4.1.8.1 linkingAgentIdentifierType (M, NR)
 - 4.1.8.2 linkingAgentIdentifierValue (M, NR)
 - 4.1.8.3 linkingAgentRole (M, NR)
- 4.2 rightsExtension (O, R)

Semantic unit	4.1 rightsStatement
Semantic components	4.1.1 rightsStatementIdentifier 4.1.2 rightsBasis 4.1.3 copyrightInformation 4.1.4 licenseInformation 4.1.5 statuteInformation 4.1.6 rightsGranted 4.1.7 linkingObjectIdentifier 4.1.8 linkingAgentIdentifier
Definition	Documentation of the repository's right to perform one or more acts.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional
Usage notes	<p>This semantic unit is optional because in some cases rights may be unknown. Institutions are encouraged to record rights information when possible.</p> <p>Either <i>rightsStatement</i> or <i>rightsExtension</i> must be present if the Rights entity is included.</p> <p>The <i>rightsStatement</i> should be repeated when the act(s) described has more than one basis, or when different acts have different bases.</p>

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.1 rightsStatementIdentifier
Semantic components	4.1.1.1 rightsStatementIdentifierType 4.1.1.2 rightsStatementIdentifierValue
Definition	The designation used to uniquely identify the rights statement within a preservation repository system.
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
Repeatability	Not repeatable
Obligation	Mandatory
Creation/ maintenance notes	The <i>rightsStatementIdentifier</i> 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.

Semantic unit	4.1.1.1 rightsStatementIdentifierType
Semantic components	None
Definition	A designation of the domain within which the rights statement identifier is unique.
Data constraint	Value should be taken from a controlled vocabulary.
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.1.2 rightsStatementIdentifierValue
Semantic components	None
Definition	The value of the <i>rightsStatementIdentifier</i> .
Data constraint	None
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	4.1.2 rightsBasis
Semantic components	None
Definition	Designation of the basis for the right or permission described in the <i>rightsStatementIdentifier</i> .
Data constraint	Values should be taken from a controlled vocabulary.
Repeatability	Not repeatable
Obligation	Mandatory
Usage notes	<p>Suggested values: copyright, license, statute.</p> <p>When <i>rightsBasis</i> is “copyright”, <i>copyrightInformation</i> should be provided.</p> <p>When <i>rightsBasis</i> is “license”, <i>licenseInformation</i> should be provided.</p> <p>When <i>rightsBasis</i> is “statute”, <i>statuteInformation</i> should be provided.</p> <p>If the basis for the rights is the item is public domain, use “copyright”. If the basis is Fair Use, use “statute”.</p> <p>If more than one basis applies, the entire rights entity should be repeated.</p>

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.3 copyrightInformation
Semantic components	4.1.3.1 copyrightStatus 4.1.3.2 copyrightJurisdiction 4.1.3.3 copyrightStatusDeterminationDate 4.1.3.4 copyrightNote
Definition	Information about the copyright status of the object(s).
Data constraint	Container
Repeatability	Not repeatable
Obligation	Optional
Usage notes	<p>When <i>rightsBasis</i> is “copyright”, <i>copyrightInformation</i> should be provided.</p> <p>Repositories may need to extend this with more detailed information. See the California Digital Library's copyrightMD schema (www.cdlib.org/inside/projects/rights/schema/) for an example of a more detailed scheme.</p>

Semantic unit	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 is recorded.
Data constraint	Values should be taken from a controlled vocabulary.
Repeatability	Not repeatable
Obligation	Mandatory
Usage notes	Suggested values: copyrighted = Under copyright. publicdomain = In the public domain. unknown = Copyright status of the resource is unknown.

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.3.2 copyrightJurisdiction
Semantic components	None
Definition	The country whose copyright laws apply.
Rationale	Copyright law can vary from country to country.
Data constraint	Values should be taken from ISO 3166.
Example	us de
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	4.1.3.3 copyrightStatusDeterminationDate
Semantic components	None
Definition	The date that the copyright status recorded in <i>copyrightStatus</i> was determined.
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.
Examples	20070608
Repeatability	Not repeatable
Obligation	Optional

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.3.4 copyrightNote
Semantic components	None
Definition	Additional information about the copyright status of the object.
Data constraint	None
Examples	Copyright expiration expected in 2010 unless renewed. Copyright statement is embedded in file header.
Repeatability	Repeatable
Obligation	Optional

Semantic unit	4.1.4 licenseInformation
Semantic components	4.1.4.1 licenseIdentifier 4.1.4.2 licenseTerms 4.1.4.3 licenseNote
Definition	Information about a license or other agreement granting permissions related to an object.
Data constraint	Container
Repeatability	Not repeatable
Obligation	Optional
Usage notes	When <i>rightsBasis</i> is “license”, <i>licenseInformation</i> should be provided.

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.4.1 licenseIdentifier
Semantic components	4.1.4.1.1 licenseIdentifierType 4.1.4.1.2 licenseIdentifierValue
Definition	A designation used to identify the granting agreement uniquely within the repository system.
Data constraint	Container
Repeatability	Not repeatable
Obligation	Optional
Usage notes	<p>This semantic unit is intended to refer to a document recording the granting of permission. 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.</p> <p>The identifier is optional because the agreement may not be stored in a repository with an identifier. In the case of a verbal agreement, for example, the entire agreement may be included or described in the <i>licenseTerms</i>.</p>

Semantic unit	4.1.4.1.1 licenseIdentifierType
Semantic components	None
Definition	A designation of the domain within which the license identifier is unique.
Data constraint	Value should be taken from a controlled vocabulary.
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.4.1.2 licenseIdentifierValue
Semantic components	None
Definition	The value of the <i>licenseIdentifier</i> .
Data constraint	None
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	4.1.4.2 licenseTerms
Semantic components	None
Definition	Text describing the license or agreement by which permission was granted.
Data constraint	None
Repeatability	Not repeatable
Obligation	Optional
Usage notes	This could contain the actual text of the license or agreement or a paraphrase or summary.

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.4.3 licenseNote
Semantic components	None
Definition	Additional information about the license.
Data constraint	None
Example	License is embedded in XMP block in file header.
Repeatability	Repeatable
Obligation	Optional
Usage notes	Information about the terms of the license should go in <i>licenseTerms</i> . <i>licenseNotes</i> 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.

Semantic unit	4.1.5 statuteInformation
Semantic components	4.1.5.1 statuteJurisdiction 4.1.5.2 statuteCitation 4.1.5.3 statuteInformationDeterminationDate 4.1.5.4 statuteNote
Definition	Information about the statute allowing use of the object.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional
Usage notes	When <i>rightsBasis</i> is “statute”, <i>statuteInformation</i> should be provided.

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.5.1 statuteJurisdiction
Semantic components	None
Definition	The country or other political body enacting the statute.
Rationale	The connection between the object and the rights granted is based on jurisdiction.
Data constraint	Values should be taken from a controlled vocabulary.
Example	us de
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	4.1.5.2 statuteCitation
Semantic components	None
Definition	An identifying designation for the statute.
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 34
Repeatability	Not repeatable
Obligation	Mandatory
Usage notes	Use standard citation form when applicable.

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.5.3 statuteInformationDeterminationDate
Semantic components	None
Definition	The date that the determination was made that the statute authorized the permission(s) noted.
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.
Examples	2007-12-01 20040223151047.0
Repeatability	Not repeatable
Obligation	Optional

Semantic unit	4.1.5.4 statuteNote
Semantic components	None
Definition	Additional information about the statute.
Data constraint	None
Example	Applicability to web-published content sent for review by general counsel 9/19/2008.
Repeatability	Repeatable
Obligation	Optional

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.6 rightsGranted
Semantic components	4.6.1 act 4.6.2 restriction 4.6.3 termOfGrant 4.6.4 rightsGrantedNote
Definition	The action(s) that the granting agency has allowed the repository.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional

Semantic unit	4.1.6.1 act
Semantic components	None
Definition	The action the preservation repository is allowed to take.
Data constraint	Value should be taken from a controlled vocabulary.
Repeatability	Not repeatable
Obligation	Mandatory
Usage notes	<p>Suggested values:</p> <p>replicate = make an exact copy</p> <p>migrate = make a copy identical in content in a different file format</p> <p>modify = make a version different in content</p> <p>use = read without copying or modifying (e.g., to validate a file or run a program)</p> <p>disseminate = create a copy or version for use outside of the preservation repository</p> <p>delete = remove from the repository</p> <p>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 <i>eventType</i>.</p>

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.6.2 restriction
Semantic components	None
Definition	A condition or limitation on the act.
Data constraint	None
Examples	No more than three Allowed only after one year of archival retention has elapsed Rightsholder must be notified after completion of act
Repeatability	Repeatable
Obligation	Optional

Semantic unit	4.1.6.3 termOfGrant
Semantic components	4.1.6.3.1 startDate 4.1.6.3.2 endDate
Definition	The time period for the permissions granted.
Rationale	The permission to preserve may be time bounded.
Data constraint	Container
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.6.3.1 startDate
Semantic components	None
Definition	The beginning date of the permission granted.
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.
Examples	2006-01-02 20050723
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	4.1.6.3.2 endDate
Semantic components	None
Definition	The ending date of the permission granted.
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.
Examples	2010-01-02 20120723
Repeatability	Not repeatable
Obligation	Optional
Usage notes	Use “OPEN” for an open ended term of grant. Omit <i>endDate</i> if the ending date is unknown or the permission statement applies to many objects with different end dates.

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.6.4 rightsGrantedNote
Semantic components	None
Definition	Additional information about the rights granted.
Rationale	A textual description of the rights granted may be needed for additional explanation.
Data constraint	None
Repeatability	Repeatable
Obligation	Optional
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.

Semantic unit	4.1.7 linkingObjectIdentifier
Semantic components	4.1.7.1 linkingObjectIdentifierType 4.1.7.2 linkingObjectIdentifierValue
Definition	The identifier on an object associated with the rights statement.
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 provides the mechanism for the link from the rights statement to an object.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional
Usage notes	<i>linkingObjectIdentifier</i> is optional because in some cases it will be more practical to link from the object(s) to the rights statement; for example, a repository may have a single rights statement covering thousands of public domain objects.

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.7.1 linkingObjectIdentifierType
Semantic components	None
Definition	A designation of the domain in which the linking object identifier is unique.
Data constraint	Value should be taken from a controlled vocabulary.
Examples	[see examples for <i>objectIdentifierType</i>]
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	4.1.7.2 linkingObjectIdentifierValue
Semantic components	None
Definition	The value of the <i>linkingObjectIdentifier</i> .
Data constraint	None
Examples	[see examples for <i>objectIdentifierValue</i>]
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.8 linkingAgentIdentifier
Semantic components	4.1.8.1 linkingAgentIdentifierType 4.1.8.2 linkingAgentIdentifierValue 4.1.8.3 linkingAgentRole
Definition	Identification of one or more agents associated with the rights statement.
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
Repeatability	Repeatable
Obligation	Optional
Usage notes	<i>linkingAgentIdentifier</i> is optional because a relevant agent may be unknown, or in no agent may be relevant. The latter is likely when the rights basis is statute.

Semantic unit	4.1.8.1 linkingAgentIdentifierType
Semantic components	None
Definition	A designation of the domain in which the linking agent identifier is unique.
Data constraint	Value should be taken from a controlled vocabulary.
Examples	[see examples for agentIdentifierType]
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	4.1.8.2 linkingAgentIdentifierValue
Semantic components	None
Definition	The value of the <i>linkingAgentIdentifier</i> .
Data constraint	None
Examples	[see examples for <i>agentIdentifierValue</i>]
Repeatability	Not repeatable
Obligation	Mandatory

Semantic unit	4.1.8.3 linkingAgentRole
Semantic components	None
Definition	The role of the agent in relation to the rights statement.
Data constraint	Values should be taken from a controlled vocabulary.
Examples	contact creator publisher rightsholder grantor
Repeatability	Not repeatable
Obligation	Mandatory

THE PREMIS DATA DICTIONARY

Semantic unit	4.2 rightsExtension
Semantic components	Defined externally
Definition	A container to include semantic units defined outside of PREMIS.
Rationale	There may be a need to replace or extend PREMIS defined units.
Data constraint	Container
Repeatability	Repeatable
Obligation	Optional
Usage notes	<p>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,” page 19.</p> <p>Either <i>rightsStatement</i> or <i>rightsExtension</i> must be present if the Rights entity is included.</p> <p>If <i>rightsExtension</i> container needs to be associated explicitly with any PREMIS subunit under <i>rights</i>, the container <i>rights</i> is repeated. If extensions from different external schemas are needed, <i>rights</i> should also be repeated.</p>

SPECIAL TOPICS

As it compiled the Data Dictionary, the PREMIS working group felt several topics were important but too detailed for the Data Dictionary itself. The discussion here provides background information about semantic units and illustrates the thinking of the working group.

Format information

The working group discussed format at length, finding a need to come to agreement on some fundamental questions before specific semantic units could be defined. These issues included:

- What is a format?
- What types of objects have 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 the group wanted to be very specific about its meaning. In discussion the defining feature of a format emerged as the fact that a format has to correspond to some formal or informal specification; it cannot be a random or undocumented layout of bits. The definition in the Wikipedia, “a particular way to encode information for storage in a computer file,” does not seem to emphasize this feature sufficiently.²⁴ The group drafted its own definition: *a specific, preestablished 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” for the more generic “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 a code or string that can in turn be associated with the full format specification is more practical. The group saw format name as a somewhat arbitrary designation that could be used as this indirect link. However, two complications arose when the group attempted 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. There was some discussion of whether the semantic unit defined for format name should include both format and version (e.g., “TIFF 6.0”) or whether two semantic units should be defined, one for name and one for version. To allow existing authority lists such as MIME type to be used the group decided on two semantic units. In the Data Dictionary *formatDesignation* has two components: *formatName* and *formatVersion*.

SPECIAL TOPICS

Second, centrally maintained format registries are expected to be the best way to get detailed format information in the future.²⁵ 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. The group discussed whether to combine all format identification into a single set of semantic units, or 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 the group was reluctant 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.

The group 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.

- (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. Repeatability of the *format* element makes it possible to capture them. *formatNote* should be used to record supplementary, qualifying information, when several identifications form a disjunction of candidate formats [e.g., TIFF 3.0 or TIFF 4.0].

It is not uncommon for particular implementations of formats to be specified, often called profiles. 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. The group discussed various options to accommodate this, such as making the format designation repeatable or defining format profile as a separate semantic unit. Instead the decision was to recommend recording the most specific format designation that applies. A repository (or formats registry) may use multipart format names (e.g., “TIFF_GeoTIFF” or “WAVE_MPEG_BWF”) to achieve this specificity.

The group recognized that the most specific designation is a matter of opinion and will be implementation specific. For example, for a METS¹⁰ 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.

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.

Since digital environments are made up of components that can be broken down into smaller and smaller components, their descriptions can easily become extremely complex. It is also possible that these descriptions will tend to be the same for entire classes of digital objects, for example, for all files of a particular format. Both of these factors suggest that the most efficient model for collecting and maintaining environment metadata is a centralized registry. While the development of the PREMIS *environment* container did not presuppose the existence of such a registry, it might best be interpreted as a template for the types of information an environment registry might maintain, rather than what a repository is likely to record locally.

The semantic units associated with the *environment* container represent the PREMIS working group's recommendation of what a repository needs to know about an archived object's environment. How this information is known—through a central registry, through locally recorded metadata, or both—is an implementation issue that must be resolved by the repository.

The working group decided to limit its scope to environment metadata associated with objects currently in the repository. Strategies for recording changes to the environment over time is an implementation issue and therefore beyond the scope of the Data Dictionary.

Sometimes multiple environments support a single digital format. The Data Dictionary acknowledges this possibility by making the *environment* container repeatable, but this is in no way intended to suggest that a repository should attempt to account for every possible software/hardware combination compatible with a particular archived object. Documented environments should, however, include the semantic unit *environmentCharacteristic*, populated by an appropriate value such as “minimum,” “recommended,” “known to work,” etc. The working group generally agreed that at least a “minimum” environment should be specified. Specification of an environment that is “known to work” may be necessary in cases where it is important to preserve certain significant properties of the object—aspects of the object's original look, feel, and functionality. In these circumstances, it is useful to document an environment that is known to support these attributes faithfully.

The working group considered whether environment metadata can usefully apply to representations, files, and bitstreams. Although in most cases it does not apply to bitstreams, since software operates on known file formats, or in the case of compound objects, on aggregations of known file formats, 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

SPECIAL TOPICS

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.

However, 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 detail separate environment information for each of these bitstreams if they are both handled by any rendering application compatible with the TIFF format specification.

Note that environment metadata may differ at the representation and file levels for a particular Object. For example, a browser is appropriate for rendering a multimedia Web page consisting of text, static images, animation, and sound components, but each component rendered separately would require different environments than the one for the compound object as a whole.

The working group decided not to recommend supplying separate environment information for both the preservation and the dissemination versions of an Object (where the dissemination version is the version made available to users in a Dissemination Information Package or DIP). If dissemination versions are stored by the repository separately from preservation masters, these are stored objects and can be described by all metadata applicable to Object entities. If dissemination versions are generated “on the fly” from stored preservation masters, the environment to support them is not strictly a preservation issue. While environment information for dissemination versions may in some cases be useful, it is not core in the sense of being necessary to support the preservation process. (See also the discussion of dissemination format, page [205](#).)

Another point of discussion was whether the mechanism(s) by which archived objects are delivered from the repository to the user (i.e., over a network, on CD, on DVD, etc.) should be part of the environment metadata. The argument in favor of this is that the rendering environment must support the requirements implied by the delivery mechanism—if content is delivered on CD-ROM, the rendering environment must include a CD-ROM drive. However, the group decided that knowledge of the delivery mechanism was not essential to support the preservation process and therefore not core. Moreover, the usefulness of a delivery mechanism description will likely vary from repository to repository, depending on local dissemination policies.

Despite the critical importance of environment metadata for ensuring that digital materials remain accessible and usable over the long term, the working group reluctantly decided to make the entire *environment* container optional. The group could not assert categorically that every preservation strategy that exists or might be developed would require knowledge of environment information. However, the fact that the *environment* container is currently optional does not indicate that the working group considers this metadata unimportant. Well-documented

environments for access and use are an essential component of most digital preservation strategies. Much work remains to be done, however, to establish practical mechanisms for collecting, storing, and updating this metadata.

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. The group arrived at 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 encoding or encryption. 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 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.

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 encryption or encoding; 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.

SPECIAL TOPICS

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 distinct metadata records 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.

Fixity, integrity, authenticity

In the process of defining core elements for preservation the working group gave 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, recommended practice is to create and test at least two message digests using two 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 determine the level of compliance to a particular format specification.²⁶ 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 appropriately named.

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.”²⁷

Authentication, or the demonstration of authenticity, is multifaceted, and includes both technical and procedural aspects. Technical approaches may include the maintenance of detailed documentation of digital provenance (the history of the object), the preservation of a version of the object that is, bit-wise, identical to the content as submitted, 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 are discussed next.

Digital signatures

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

SPECIAL TOPICS

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. 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

For a preservation repository to later validate a digital signature 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 reimplemented 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.²⁸ 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 PREMIS model) 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:

- 1.9 signatureInformation (O, R) [file, bitstream]
 - 1.9.1 signature (O, R)
 - 1.9.1.1 signatureEncoding (M, NR) [file, bitstream]
 - 1.9.1.2 signer (O, NR) [file, bitstream]
 - 1.9.1.3 signatureMethod (M, NR) [file, bitstream]
 - 1.9.1.4 signatureValue (M, NR) [file, bitstream]
 - 1.9.1.5 signatureValidationRules (M, NR) [file, bitstream]
 - 1.9.1.6 signatureProperties (O, R) [file, bitstream]
 - 1.9.1.7 keyInformation (O, NR) [file, bitstream]
 - 1.9.2 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. Repositories are encouraged to use “KeyInfo” definitions where they apply.

The semantic units discussed above have analogs in the *XMLDsig*:

PREMIS	XMLDsig
signatureMethod	<SignedInfo><SignatureMethod>
signatureValue	<SignatureValue>
signatureProperties	<Object><SignatureProperties>
keyInformation	<KeyInfo>

Three semantic units not included in *XMLDsig* were added to the Data Dictionary: *signatureEncoding*, *signer*, and *signatureValidationRules*. The semantic unit *signatureEncoding* 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

SPECIAL TOPICS

mandatory in *XMLDsig* (*signatureMethod*, *signatureValue*) or implied by the requirements of the *XMLDsig* specification (*signatureEncoding*, *signatureValidationRules*). In cases where a repository can not 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

The working group decided not to include some metadata concepts 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 will be documented in *objectCharacteristics*. 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: The *Framework* defines “quirks” as “any loss in functionality or change in the look and feel of the Content Data Object resulting from the preservation processes and procedures implemented by the archive.” The working group used “anomalies” to describe aspects of an object that do not meet the specification for the object. The discussions of quirks and anomalies centered on whether they should be defined as 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 treats quirks and anomalies as outcomes of events, recorded in *eventOutcomeDetail*.

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. The working group decided that byte order might better be treated as format-specific technical metadata, and noted that ANSI/NISO Z39.87 (Data Dictionary – Technical Metadata for Digital Still Images)¹² includes byte order as technical metadata for images.

Character encoding: This element is important, but it is format-specific technical metadata, useful only for text files and files that can include text.

Dissemination format: A great deal of discussion centered on whether dissemination format was in scope. The working group concluded 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 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. Although the group agreed that dissemination format is important to a repository operationally, it is not core to preservation processes.

Embedded metadata: One implementation used a metadata flag to indicate whether a file object contained embedded metadata. The group agreed to leave this indicator out of the Data Dictionary 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.

The group also discussed the 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. Among these, the group agreed that microfilming (preservation reformatting), moving a file offline, and media refreshment were 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 was 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 usually sufficient for retrieving the file. Therefore, a broader, less system-dependent semantic unit was defined: *contentLocation* can be interpreted narrowly (a value could be an exact path or a “fully qualified” path or

SPECIAL TOPICS

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: The *Framework* included a “Global Identifier” defined as an identifier known outside of the repository system. The group did 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. The issue was raised 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.

The *Framework* also implied that a Global Identifier would be a standard identifier such as ISBN or ISSN. However, because these schemes designate an abstract bibliographic entity or set of items, not the specific content data object in the preservation repository, they are really descriptive metadata rather than preservation metadata. ISBNs, ISSNs, and similar standard identifiers are likely to refer to many different representations held in many different preservation repositories, with no way to distinguish between them. Therefore, the identifier used by the repository must in practice be the “global” identifier.

MIME type: The Internet Media Type and SubType (commonly called “MIME type”) was subsumed under *formatIdentification*. Format identification 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 object at any given time. If an object is changed or modified, a new object is created that is related to the previous one. Each object then has its own set of metadata, and the relationship between the two is also described. The model does not allow for modifying an object and keeping a set of metadata that describes a history of changes about that object. Therefore, there would be no modification date of an 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. Modification date was considered by the group 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.

Object type: The group discussed the desirability of having a semantic unit for a genre or media type that would classify objects on a much higher level than format. There is such an element in the METS schema, but currently there is no controlled vocabulary defined for its value. The group argued that object type is 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. High-level object typing is probably more useful for exchange and access to objects than for preservation purposes. However, developing a universally acceptable list of object types is beyond the PREMIS’s scope

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 group discussed how the National Library of Medicine’s Permanence ratings intersected with PREMIS work.²⁹ The permanence-level rating appeared to be less a property of an Object entity than a property of an entity defining business rules. The group had already decided that business rules were out of scope.

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 decision to recommend recording only a single format at the most specific level obviated the need to define a separate semantic unit for profile conformance.

Reason for creation: This metadata element was defined in the *Framework*. The working group concluded that for objects created by the preservation repository (e.g., a normalized version of a file) the reason for creation could be recorded as part of the *eventDetail* for the event of creation. However, the group did not consider at length events or processes that occur before ingest and was not convinced that these were core knowledge for a preservation repository. Some of the context surrounding object creation may be documented in relation to the Object entity in *creatingApplication*. The group expressed some reservations about the life-cycle model used by the *Framework* (origin, pre-ingest, ingest, archival retention, etc.) as being too restrictive.

Sibling relationships: The group discussed whether sibling relationships (children of the same parent) should be made a separate category of relationship. It was agreed that sibling relationships always have a structural relationship (and may possibly also have a derivation relationship), and should therefore fall under these relationship categories. 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.

METHODOLOGY

The Core Elements Subgroup began by analyzing the Preservation Description Information recommendations of the earlier Preservation Metadata Framework working group. In OAIS, Preservation Description Information includes reference information (identifiers and bibliographic information), context information (how objects are related to each other), provenance information (the history of digital content), and “fixity” information. Members of the subgroup from institutions actively running or developing preservation repositories mapped elements from the *Framework* to those in use in their own systems. The subgroup also reviewed published specifications from organizations and projects that did not have representatives on the PREMIS working group.

It became clear that the prototype elements detailed in the *Framework* did not always correspond to elements implemented in practice. However, the exercise provided a common denominator for diverse implementations; the group discussed each element in conference calls to discover commonality in usage. Widely used elements formed the beginning of a set of core elements, which were then mapped to appropriate entity types as the data model evolved.

In the OAIS and the *Framework*, technical metadata is considered Representation Information rather than Preservation Descriptive Information. Because there are few technical metadata elements in the *Framework*, the working group compiled a list of potential technical metadata based on specifications for the proposed Global Digital Format Registry (GDFR), supplemented by data elements used in the repository systems of members’ institutions.³⁰ Each element on the list was then discussed at some length, and any element that was format specific or implementation specific was regarded as non-core. In some cases outside experts were asked to help with particularly difficult areas, including formats, hardware and software environment information, and digital signatures.

The process for determining which semantic units were core involved analysis and discussion of a selection of elements from various sources and a determination of whether they were in scope. In general, the working group excluded these candidates from the Data Dictionary:

- Metadata elements that could be grouped into broader categories.
- Format-specific, implementation-specific, or policy-driven elements.
- Elements outside the PREMIS scope.
- Elements for which information could be obtained easily and reliably from the object itself or other sources.

GLOSSARY

Early in its work, the PREMIS working group realized the need for a glossary, since a common vocabulary seemed to be lacking in discussions about preservation metadata. This glossary defines a number of terms used in this report; the working group recognizes that in 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](#) associated with 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 [File](#) without 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: foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=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](#).

GLOSSARY

Compound Object: [Digital Object](#) composed of multiple [Files](#), for example, a [Web Page](#) composed 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-length encoding 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: foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=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](#).

Decompression: Process of reversing the effects of data [Compression](#). (From FOLDOC: foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?decompress.)

Decryption: Process of employing any procedure used in cryptography to convert ciphertext (encrypted data) into plaintext.
(From FOLDOC: foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?decryption.)

Deletion: Process of removing a [Digital Object](#) from repository storage.

Dependency Relationship: Relationship where one [Digital Object](#) requires another [Digital Object](#) to support its function, delivery, or coherence of 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, for example, for the DVD version of a video recording). (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 a [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 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: www.bbn.com/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. [Validation](#) confirms 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.

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: www.dpconline.org/graphics/intro/definitions.html.)

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: foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=encryption.)

Entity: Abstraction for a set of “things” (agents, events, etc.) described by the same properties. The PREMIS data model defines five types of Entities: [Intellectual Entities](#), [Objects](#), [Agents](#), [Rights](#), and [Events](#).

Event: 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.

GLOSSARY

File: Named and ordered sequence of [Bytes](#) that is known by an operating system. A File can be zero or more [Bytes](#), 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 that a [Digital Object](#) has not been 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, preestablished 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.

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 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 as 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: www.bbn.com/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 Object](#) residing in a [Preservation Repository](#). See also [Fixity Check](#).

Migration: Preservation strategy in which a [Transformation](#) creates a version of a [Digital Object](#) in 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: foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?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.

GLOSSARY

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: 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](#) instantiating or embodying an [Intellectual Entity](#). A Representation is the set of stored [Files](#) and [Structural Metadata](#) needed to provide a complete and reasonable rendition of the [Intellectual Entity](#).

Rights: Assertions of one or more rights 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: 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 [Transformation](#) include [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: “Page” of 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: en.wikipedia.org/wiki/Web_page)

Web Site: A collection of [Web Pages](#), that is, HTML/XHTML documents accessible via HTTP on the Internet; 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: en.wikipedia.org/wiki/Web_page)

NOTES

- ¹ *A Metadata Framework to Support the Preservation of Digital Objects* (Dublin, Ohio: OCLC Online Computer Library Center, 2002), www.oclc.org/research/projects/pmwg/pm_framework.pdf.
- 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), www.oclc.org/research/projects/pmwg/surveyreport.pdf.
- ³ *Reference Model for an Open Archival Information System (OAIS)* (Washington, DC: Consultative Committee for Space Data Systems, 2002), public.ccsds.org/publications/archive/650x0b1.pdf.
- 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), www.natlib.govt.nz/files/4initiatives_metaschema_revised.pdf.
- 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.
- ⁶ IFLA, *Functional Requirements for Bibliographic Records* (Munich: K.G. Saur, 1998), www.ifla.org/VII/s13/frbr/frbr.pdf.
- ⁷ Coyle, Karen, *Rights in the PREMIS Data Model*, <http://www.loc.gov/standards/premis/Rights-in-the-PREMIS-Data-Model.pdf>.
- ⁸ Hirtle, Peter B., *Digital Preservation and Copyright*, http://fairuse.stanford.edu/commentary_and_analysis/2003_11_hirtle.html.
- ⁹ California Digital Library, copyrightMD schema, <http://www.cdlib.org/inside/projects/rights/schema/>.
- ¹⁰ Metadata Encoding & Transmission Standard (METS), <http://www.loc.gov/standards/mets/>.
- ¹¹ The Dublin Core Metadata Element Set, <http://www.dublincore.org/documents/dces/>.
- ¹² *Data Dictionary—Technical Metadata for Digital Still Images*, ANSI/NISO Z39.87-2006, http://www.niso.org/standards/standard_detail.cfm?std_id=731.
- ¹³ *Information technology – Multimedia framework (MPEG-21)*, ISO/IEC 21000 (multiple parts), International Organization for Standardization.
- ¹⁴ Synchronized Multimedia Integration Language (SMIL), <http://www.w3.org/TR/REC-smil/>.
- ¹⁵ Resource Description Framework (RDF), www.w3.org/RDF/.
- ¹⁶ MARC 21, <http://www.loc.gov/marc/>.

-
- ¹⁷ Metadata Object Description Schema (MODS), <http://www.loc.gov/standards/mods/>.
- ¹⁸ *Content Standard for Digital Geospatial Metadata*, FGDC-STD-001-1998, <http://www.fgdc.gov/metadata/csdgm/>.
- ¹⁹ VRA Core 4.0, <http://www.vraweb.org/projects/vracore4/>.
- ²⁰ Encoded Archival Description (EAD), <http://www.loc.gov/ead/>.
- ²¹ Data Documentation Initiative (DDI), <http://www.ddialliance.org/>.
- ²² vCard, <http://www.imc.org/pdi/>.
- ²³ Metadata Authority Description Schema (MADS), <http://www.loc.gov/standards/mads/>.
- ²⁴ Wikipedia, the free encyclopedia, en.wikipedia.org/wiki/Main_Page.
- ²⁵ See, for example, the proposed Global Digital Format Registry at hul.harvard.edu/gdfr/.
- ²⁶ JHOVE – JSTOR/Harvard Object Validation Environment, hul.harvard.edu/jhove/.
- ²⁷ Digital Preservation Coalition Handbook, www.dpconline.org/graphics/intro/definitions.html.
- ²⁸ *XML-Signature Syntax and Processing*, W3C Recommendation 12 February 2002, www.w3.org/TR/xmlsig-core/.
- ²⁹ Margaret Byrnes, *Assigning Permanence Levels to NLM's Electronic Publications* (presented at 2000 Preservation: An International Conference on the Preservation and Long Term Accessibility of Digital Materials), www.rlg.org/en/page.php?Page_ID=244.
- ³⁰ Global Digital Format Registry (GDFR) Data Model, <http://hul.harvard.edu/gdfr/documents.html#data>.