Understanding PREMIS

Library of Congress Network Development and MARC Standards Office

Author: Priscilla Caplan

Revision: PREMIS Editorial Committee

Originally issued in 2009; revised in 2017

Table of Contents

UNDERSTANDING PREMIS

This guide is a relatively brief overview of the PREMIS preservation metadata standard. It will not give you enough information to implement PREMIS, but it will give you some idea of what PREMIS is all about. For many readers, this will be enough. For those who do need to tackle the 250+ page *PREMIS Data Dictionary for Preservation Metadata*, this guide may serve as a gentle introduction that makes the larger document feel more familiar.

1. PREMIS IN CONTEXT

1.1. What is preservation metadata?

If you work in a library or archives or museum, chances are good you know at least something about metadata and resource description. You probably know that metadata is categorized according to what it is intended to accomplish: descriptive metadata helps in discovery and identification of resources, administrative metadata helps in managing and tracking them, and structural metadata indicates how complex digital objects are put together so that they can be properly rendered. Similarly, *preservation metadata* supports activities intended to ensure the long-term usability of a digital resource.

The PREMIS Data Dictionary defines preservation metadata as "the information a repository uses to support the digital preservation process." Here are some examples of preservation activities and how metadata can support them:

- A resource must be stored securely so that nobody can modify it inadvertently (or maliciously). Checksum information stored as metadata can be used to tell if a stored file has changed between two points in time.

- Files must be stored on media that can be read by current computers. If the media are damaged or obsolete (like the 8" floppy disks used in the 1970s) it can be difficult or impossible to recover the data. Metadata can support media management by recording the type and age of storage media and the dates that files were last refreshed.

- Over long periods of time even popular file formats can become obsolete, meaning no current applications can render them. Preservation managers must employ *preservation strategies* to ensure the resources remain usable. This might mean migrating old formats to newer equivalents, or emulating the old rendering environment on newer hardware and software. Both *migration* and *emulation* strategies require metadata about the original file formats and the hardware and software environments supporting them.

- Preservation strategies may entail changing original resources (migration) or changing how they are rendered (emulation). This can put the authenticity of the resource in doubt. Metadata can help support authenticity by documenting the *digital provenance* of the resource -- its chain of custody and authorized change history.

1.2. What is PREMIS?

PREMIS stands for "PREservation Metadata: Implementation Strategies" which is the name of an international working group sponsored by OCLC and RLG from 2003-2005. That working group produced a report called *PREMIS Data Dictionary for Preservation Metadata* which includes both a data dictionary and quite a bit of narrative about preservation metadata. The Library of Congress then published a set of PREMIS schemas for representing metadata elements in the Data Dictionary in XML. An updated second edition of the Data Dictionary and a new supporting schema were issued in March 2008, followed by minor revisions, versions 2.1 and 2.2. In June 2015 a major revision, version 3.0, was issued.

There is an active PREMIS Maintenance Activity sponsored by the Library of Congress. This includes a website linking to all sorts of official and unofficial PREMIS information, a discussion list and wiki for PREMIS implementers, and an Editorial Committee responsible for revisions to the Data Dictionary and schema. The Maintenance Activity also tries to promote awareness of PREMIS, sponsors tutorials and implementation meetings about using PREMIS, and commissions studies and publications related to PREMIS, like this guide.

Usually, when people refer to "PREMIS" they generally mean the Data Dictionary. Occasionally they may be referring to the XML schema, to the working group, or to the entire effort including the Maintenance Activity.

PREMIS Data Dictionary: www.loc.gov/premis/v3/premis-3-0-final.pdf
PREMIS Website: www.loc.gov/premis
PREMIS Implementers Group discussion list: pig@loc.gov
to subscribe send mail to listserv@loc.gov with the message:
subscribe pig [Your Name]

1.3. What is in the PREMIS Data Dictionary?

The PREMIS Data Dictionary defines a core set of metadata elements (actually "semantic units," but we'll talk about that later) that repositories should know in order to perform their preservation functions. Preservation functions can vary from one repository to another, but will generally include actions to ensure that digital objects remain viable (i.e., can be read from media) and renderable (i.e., can be displayed, played or otherwise interpreted by application software), as well as to ensure that digital objects in the repository are not inadvertently altered, and that legitimate changes to objects are documented.

The Data Dictionary is not intended to define all possible preservation metadata elements, only those that most repositories will need to know most of the time. Several categories of metadata are excluded as out of scope, including:

- Format-specific metadata, i.e., metadata that pertains to only one file format or class of formats such as audio, video or vector graphics.
- Implementation-specific metadata and business rules, i.e., metadata that describes the policies or practices of an individual repository, such as how it provides access to materials.
- Descriptive metadata. Although resource description is obviously relevant to preservation, many independent standards can be used for this purpose, including MARC, MODS, and Dublin Core.

- Detailed information about media or hardware. Again, although clearly relevant to preservation, this metadata is left to other communities to define.

- Detailed information about agents (people, organizations or software) other than what is needed for identification.

- Extensive information about rights and permissions; the focus is on those that affect preservation functions.

If you think of all of the metadata needed by an organization running a preservation repository, PREMIS can be seen as defining a subset in the center. On the one hand, it is not concerned with discovery and access, and on the other, it does not attempt to define detailed format-specific metadata. It defines only that metadata commonly needed to perform preservation functions on all materials.

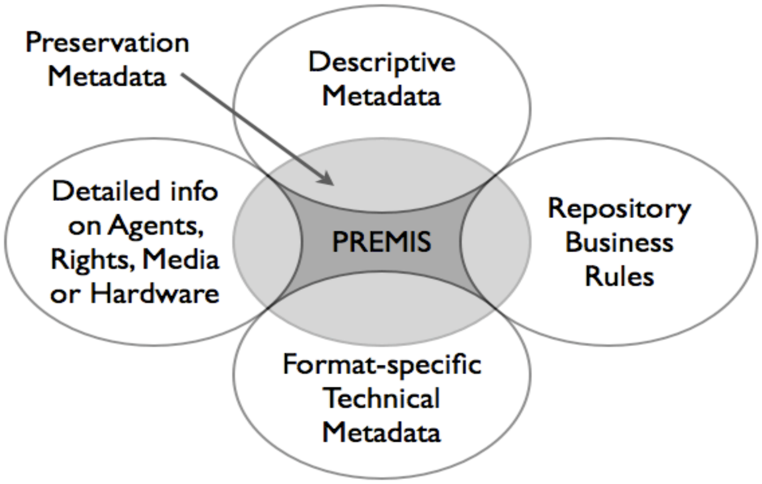

Preservation Metadata
Descriptive Metadata
Detailed info on Agents, Rights, Media or Hardware
PREMIS
Repository Business Rules
Format-specific Technical Metadata

Figure 1: PREMIS as a subset of all Preservation Metadata

Figure 1 shows all preservation metadata as the circle in the center of the diagram. It includes some descriptive metadata, some business rules, some detailed technical metadata, and some detailed information about agents, rights, media and hardware. PREMIS is the small core at the heart of preservation metadata that excludes all these other types.

1.4 How should PREMIS be used?

The PREMIS Data Dictionary defines what a preservation repository needs to know. It is important to note that the focus is on the repository system and its management, not on the authors of digital content, people who scan or otherwise convert analog content to digital, or staff who evaluate and license commercial electronic resources. The primary uses of PREMIS are for repository design, repository evaluation, and exchange of archived information packages.

Those designing and/or developing preservation repository software applications should use PREMIS as a guideline for what information should be obtained and recorded by the application or otherwise known to repository management.

Those who are planning to implement a preservation repository should use PREMIS as a checklist for evaluating candidate software. Systems which can support the PREMIS Data Dictionary will be better able to preserve information resources in the long term.

A working repository will sometimes want to export stored information packages for ingest into another repository. For example, a custodial organization may be migrating from one repository system to another, a customer may want to switch from one third-party service to another, or an institution may use another institution's preservation services in a trusted digital repository. PREMIS provides a common set of data elements that can be understood by both the exporting and importing repository, especially if the PREMIS XML schema is used.

1.5. Should you be using PREMIS?

It depends. Most of the staff in libraries, archives, museums and other cultural heritage organizations don't have any direct involvement in digital preservation. In that case, it's enough that you know what PREMIS is: a data dictionary for preservation metadata. If your job involves some responsibility for any aspect of digital preservation, you will probably find it useful to be familiar with PREMIS. If you are involved with the evaluation or implementation of an institutional repository or preservation system, you should have a good understanding of PREMIS. Consider making use of the training materials on the PREMIS website or taking a PREMIS tutorial when they are offered.

If you work on digitization projects you might be wondering if you should be creating PREMIS metadata for later use. Most of the PREMIS elements are designed to be automatically supplied by the preservation repository application. (Of course this does not mean that all currently available applications do supply them.) However, there is some information you should record if possible:

Inhibitors. *Inhibitors* are defined as any features of an object intended to inhibit access, use, or migration. Inhibitors include password protection and encryption. Repository software may not be able to identify an inhibitor because the inhibitor may prevent the software from analyzing the object, so if you know a file has inhibitors, it is important to record them. PREMIS defines semantic units for inhibitor type, target (the actions that are inhibited), and key (password or other mechanism to bypass the inhibitor).

Provenance. *Digital provenance* is the record of the chain of custody and change history of a digital object. If your institution created the object, the circumstances of its creation are obviously an important part of its provenance. The name and version of the creating application and the creation date can often be extracted from the file header, but not always, so recording this information is recommended. PREMIS allows change history to be recorded as Event information, which is described below. A controlled vocabulary for event types is available at id.loc.gov/vocabulary/preservation/eventType.

Many of the PREMIS event types are designed to describe actions that happen after something is submitted for ingest to a repository, although some are used for events that happen before ingest, such as capture and accession.

Significant Properties. *Significant properties* are characteristics of an object that should be maintained through preservation actions. For example, if you have a document, is it only the words and images that are critical, or are the fonts, background, formatting, and other "look and feel" features equally important? The idea of significant properties is one of the most important concepts in digital preservation and one of the least understood. Nonetheless, any institution creating or acquiring digital materials for a user community should think hard about what features of those materials are important to that community and try to record this information as significant properties.

Rights. Rights information isn't unique to preservation, of course, but knowing what you can do with an object is very important to the preservation process. Any known rights information, including copyright status, license terms and special permissions, should be recorded.

2. KEY CONCEPTS IN PREMIS

2.1. Semantic Units

The PREMIS Data Dictionary defines *semantic units*, not metadata elements. The distinction is subtle but real. A semantic unit is a piece of information or knowledge. A metadata element is a defined way of representing that information in a metadata record, schema or database. PREMIS does not specify how metadata should be represented in any system; it only defines what the system needs to know and should be able to export to other systems. So, to be a PREMIS purist, you have to think in terms of semantic units. For the rest of us, metadata elements are close enough.

The names of PREMIS semantic units are "camel case" strings. That is, words are not separated by spaces but begin with capitals: objectIdentifier, relatedEventIdentification. In this document they are printed in the Verdana font.

2.2. Containers and Subunits

Some semantic units are defined as *containers*, which means they don't hold a value themselves but exist to group related semantic units. For example, whenever you record an identifier in PREMIS you have to say what kind of identifier it is (e.g. "DOI", "ISBN", "local system assigned"). The container objectIdentifier is used to group the two subunits objectIdentifierType and objectIdentifierValue.

Containers provide a hierarchical structure to the Data Dictionary which is reflected in the numbering of semantic units:

```
1.1 objectIdentifier (M, R)
        1.1.1 objectIdentifierType (M, NR)
        1.1.2 objectIdentifierValue (M, NR)
```

This excerpt from the Data Dictionary shows at a glance that the mandatory (M) and repeatable (R) element objectIdentifier does not hold a value itself but serves as a container for the component elements objectIdentifierType and objectIdentifierValue. Because type and value are non-repeatable (NR) within the container, you would have to repeat the entire container structure to record two different identifiers.

2.3. Extension Containers

An *extension container* is a special type of container that has no subunits defined under it. It is designed to give a place for non-PREMIS metadata to be recorded. In this way, PREMIS can be extended to include metadata that is out of scope or otherwise not included in the Data Dictionary. Extension containers have "Extension" as the last part of their names.

For example, format-specific technical metadata is not included in PREMIS because it is considered out-of-scope, but is very important information for digital preservation. The extension container objectCharacteristicsExtension gives a place to record technical metadata defined by other schemas, such as the Z39.87 data dictionary for bitmap images (its XML schema is referred to as "NISO Metadata for Images in XML Schema--MIX").

If you are familiar with XML it will be obvious to you that the PREMIS Data Dictionary was designed to be compatible with XML. PREMIS semantic units can be implemented as XML elements; container units are elements that take only other elements as content, and extension units are containers for elements defined by external schema. There is more on PREMIS and XML in section 5.1 below. Later efforts produced a PREMIS OWL Ontology to be used for Linked Data applications. The first version of the ontology was compatible with PREMIS version 2.2, and, as of this writing, a revision is almost complete for version 3.0. Information about the PREMIS OWL Ontology is available at www.loc.gov/standards/premis/ontology

3. PREMIS DATA MODEL

One of the main principles behind PREMIS is that you need to be very clear about what you are describing. In versions 1 and 2 PREMIS defined five kinds of things (called *Entities*) you can talk about: Intellectual Entity, Object, Agent, Event and Rights.

In versions 1 and 2 PREMIS defined five kinds of things (called *Entities*) you can talk about: Intellectual Entity, Object, Agent, Event and Rights. *(A diagram of the data model for versions 1 and 2 is found on the PREMIS web site.)* In version 3, which was issued in June 2015, the PREMIS Data Model was revised to make Intellectual Entity another category of Object.

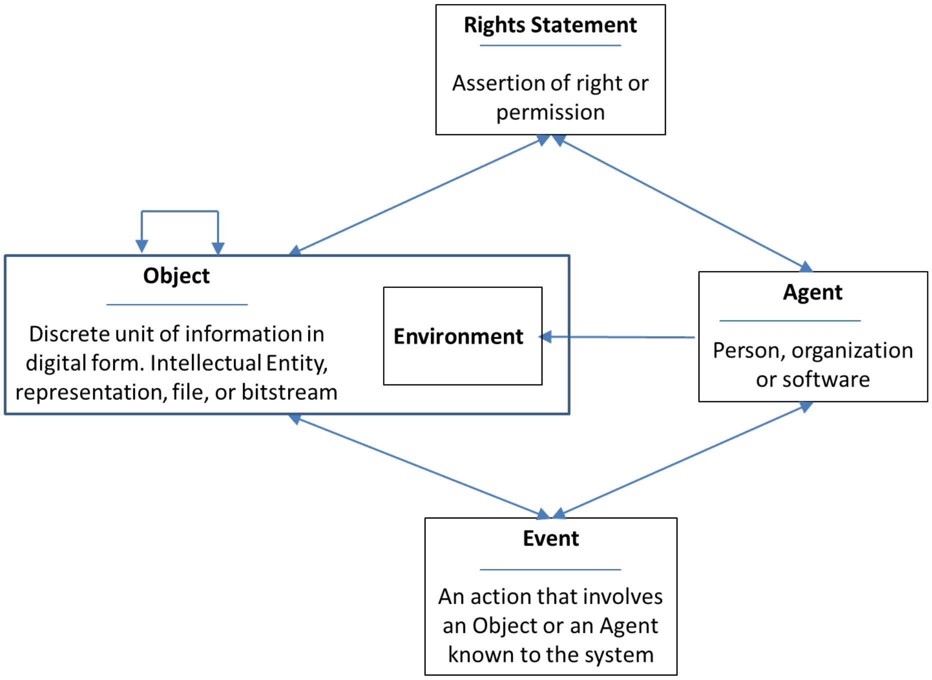

Rights Statement
Assertion of right or permission
Object
Discrete unit of information in digital form. Intellectual Entity, representation, file, or bitstream
Environment
Agent
Person, organization or software
Event
An action that involves an Object or an Agent known to the system

Figure 2: PREMIS Data Model, version 3

3.1. Object Entity

Objects are what are actually stored and managed in the preservation repository. Most of PREMIS is devoted to describing digital objects. The information that can be recorded includes:

- a unique identifier for the Object (type and value),
- fixity information such as a checksum (message digest) and the algorithm used to derive it,
- the size of the Object,
- the format of the Object, which can be specified directly or by linking to a format registry,
- the original name of the Object,
- information about its creation,
- information about inhibitors,
- information about its significant properties,
- information about its environment (see below),
- where and on what medium it is stored,
- digital signature information,
- relationships with other Objects and other types of Entities.

PREMIS actually defines four different kinds of Objects and requires implementers to make a distinction between them. These are *Bitstreams, Files*, *Representations*, and *Intellectual Entities*.

A ***File Object*** is just what it sounds like, a computer file, like a PDF or JPEG.

Bitstream Objects are subsets of files. A Bitstream Object is defined as data (bits) within a file that a) have common properties for preservation purposes, and b) cannot stand alone without adding a file header or other structure. So for example, if you had a file in AVI (audio-video interleaved) format, you might want to distinguish the audio bitstream from the video bitstream, and describe them as separate Bitstream Objects.

A ***Representation Object*** is the set of all File Objects needed to render an Intellectual Entity. For example, say you want to preserve a Web page, perhaps your institution's home page as of some date. Chances are good that the home page you see in your browser is actually composed of many different files – one or more HTML files, a handful of GIF or JPEG images, maybe a little audio or Flash animation. It probably also uses a stylesheet to create the display you see. It takes all of these files together for a browser to render the home page for viewing, so if a repository wants to preserve a renderable home page, it has to know about all these files and how to put them together. The Representation Object allows the repository not only to identify the set of related files, but also to describe characteristics of the totality (e.g. the Web page as a whole) that may be different from any of its parts.

An ***Intellectual Entity Object*** is defined as 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. PREMIS does not generally define descriptive metadata pertaining to Intellectual Entities because there are plenty of descriptive metadata standards to choose from. In versions 1 and 2 Intellectual Entities could only be referenced with an identifier because they were considered only conceptual. But in version 3 an Intellectual Entity can be described with descriptive metadata outside of PREMIS **or** with preservation metadata as an Object within PREMIS. In most cases the semantic units used for Intellectual Entities are the same as those for Representations.

PREMIS says that an Object in a preservation system should be associated with the conceptual Intellectual Entity it represents by including an identifier of the Intellectual Entity in the metadata for the Object. So, for example, if we were preserving a copy of *Buddhism: The Ebook: an Online Introduction* we might use the ISBN as the link to the Intellectual Entity description in the PREMIS description of the ebook.

Some semantic units defined in the PREMIS Data Dictionary are applicable to all four types of Object, while others are applicable to only one or two types of Object.

Several semantic units are defined to record the *environment* of an Object, that is, what hardware and software are required to render it and what dependencies there are on other Objects. In versions 1 and 2 that information is part of the Object description. In version 3 the Environment, as a special kind of Intellectual Entity Object, may be linked to from the Files, Representations and Bitstreams that use it. For example, a PDF file can be displayed by several versions of Adobe Acrobat and Adobe Reader as well as by other open source and commercial programs. Each of these, in turn, is supported on various operating systems and requires certain minimum hardware specifications (processor speed, memory and disk). Because Adobe Reader is not a standalone application but a browser plug-in, it is also dependent on certain versions of certain browsers; for example, Reader 9 for Mac OS requires the Safari browser version 2.0.4 or later. Environment information is critical to certain preservation strategies, but it is difficult and time-consuming to determine, and may best be recorded in central registries like the PRONOM registry maintained by The National Archives of the UK. PREMIS allows repositories to link to information stored in external registries when this is preferable to storing it locally.

To illustrate the relationships between Objects, consider again the Web page mentioned above. The Web page is an *Intellectual Entity* that can be expressed in a number of different *Representations*. One Representation, as detailed above, consists of a number of separate File Objects (HTML page, images, stylesheet, etc.). However, the Web page may also be archived as a single Web Archive (WARC) file (see www.digitalpreservation.gov/formats/fdd/fdd000236.shtml for details). This is a different Representation of the same Intellectual Entity, containing just a single file, as illustrated:

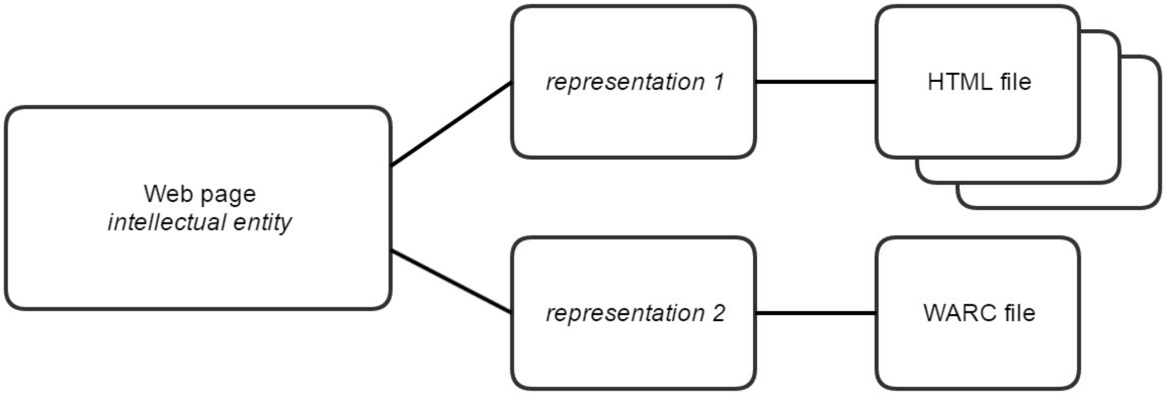


Figure 3: PREMIS Object relationship example

3.2. Events

The *Event entity* aggregates information about actions that affect Objects in the repository. An accurate and trustworthy record of Events is critical for maintaining the digital provenance of an Object, which in turn is important in demonstrating the authenticity of the Object.

The information that can be recorded about Events includes:

- a unique identifier for the Event (type and value),
- the type of Event (creation, ingestion, migration, etc.),
- the date and time the Event occurred,
- a detailed description of the Event,
- a coded outcome of the Event,
- a more detailed description of the outcome,
- Agents involved in the Event and their roles,
- Objects involved in the Event and their roles.

Each repository system must make its own decisions about which Events to record as a permanent part of an Object's history. PREMIS recommends that actions that change an Object should always be recorded, and it provides a controlled vocabulary of important Event types to encourage repositories to record these Events consistently.

This vocabulary is at http://id.loc.gov/vocabulary/preservation/eventType.

3.3. Agents

Agents are actors that have roles in Events, Rights statements (see 3.4 Rights), and Environment Objects. Agents can be people, organizations, software applications, or hardware. PREMIS defines only a minimum number of semantic units necessary to identify Agents, since there are several external standards that can be used for more detailed information. A repository could choose to use a separate standard for recording additional information about Agents, or it could use the Agent identifier to point to externally recorded information.

The Data Dictionary includes:

- a unique identifier for the Agent (type and value),
- the Agent's name,
- designation of the type of Agent (person, organization, software, hardware).
- Agent version (for software or hardware),
- a general note about the Agent,
- Events associated with the Agent,
- Rights statement associated with the Agent,
- Environment Objects associated with the Agent

Whenever an Agent is referenced in relation to an Event or a Rights statement, the role of the Agent should also be recorded. Any one Agent can have many roles. For example, I could be the author and rights holder of one work, the author (but not rights holder) of a second work, and the depositor of a third work. In the PREMIS model a repository would assign a unique identifier to me and would reference that identifier in any event record or rights statement in which I was an agent, along with my role in that particular context.

3.4. Rights

Most preservation strategies involve making identical copies and derivative versions of digital objects, actions that may be restricted by copyright law to the rights holders. The *Rights entity* aggregates information about rights and permissions pertaining to objects in a preservation repository so that the repository can do what it needs to do to preserve them. Each PREMIS Rights statement asserts two things: acts that the repository has a right to perform, and the basis for claiming that right.

For example, a repository might hold a scanned version of a book that was published in 1848 and is therefore in the public domain. The repository can do anything with its digital version based on the item's copyright status. Another repository holds an object copied from a published CD, where the shrink-wrap license allows making backup copies but restricts access and use.

The information that can be recorded in a Rights statement includes:

- a unique identifier for the Rights statement (type and value),
- whether the basis for claiming the right is copyright, license, statute, or other (e.g. institutional policy),
- more detailed information about the copyright status, license terms, or statute, as applicable,
- the action(s) that the Rights statement allows,
- any restrictions on the action(s),
- the term of grant or restriction, or time period in which the statement applies,
- the Object(s) to which the statement applies,
- Agents involved in the Rights statement and their roles.

Most of the information is designed to be *actionable* (that is, recorded in a controlled form that can be acted upon by computer program). The PREMIS Rights statement is an assertion of rights, not a record of information from which rights can be determined. That is, PREMIS does not define the kind of detailed information about authors, date and place of publication, and copyright notification that is defined, for example, in the California Digital Library's copyrightMD specification (www.cdlib.org/inside/projects/rights/schema). The purpose of copyrightMD is to help humans make rights determinations on an ongoing basis, while the purpose of the PREMIS Rights entity is to provide actionable information to preservation repository systems.

4. THE DATA DICTIONARY

4.1. Sample Data Dictionary Entry for a Simple Semantic Unit

Table 1 shows the Data Dictionary entry for the semantic unit size, which is a component or subunit of the container called objectCharacteristics. Size itself has no subunits. The Data Dictionary entry includes a definition of the element and a reason (rationale) for including it among the core PREMIS metadata, as well as examples and notes about how the value might be obtained and used. These are all intended to help implementers use the element properly.

The two rows "Object Category" and "Applicability" are used together to show whether the semantic unit is appropriate for describing Intellectual Entities, Representations, Files, and/or Bitstreams. Here size is shown to pertain to Files and Bitstreams only. Finally, there are a set of rules for use: "Data constraint," "Repeatability" and "Obligation."

Data constraints specify restrictions on the values that a semantic unit can take. In this example, the value of size must be an integer. Another common data constraint is that the value must be taken from a controlled vocabulary. If an established controlled vocabulary exists, it is referenced in data constraint, and some of the terms in the vocabulary are specified as examples in the Data Dictionary:

http://id.loc.gov/preservationdescriptions/

Other controlled vocabularies may be used, in which case the name of the vocabulary used must be recorded. There are no semantic units defined in the Data Dictionary for vocabulary names, but the PREMIS XML schema provides a place for them and, if they are identified by a URI, that will generally reflect the name of the controlled vocabulary.

Repeatability indicates whether the semantic unit can be repeated.

Obligation indicates whether a value for the semantic unit is mandatory (required) or optional. Obligation is potentially confusing, because PREMIS states clearly that it does not require a repository to store any particular information. A semantic unit that is mandatory does not have to be recorded and stored within the repository. However, the repository does have to be able to generate the value of the semantic unit when needed, such as for exchange with another repository. For example, in the unlikely event a repository stored nothing but TIFF 6.0 images, it would not have to record format information for every object. Nonetheless, the repository would know its File Objects were TIFF 6.0 images, and could provide that information if it had to. (See section 5.2. PREMIS Conformance.). Some semantic units are mandatory within a container; if that container isn't mandatory, the semantic unit will not be used if the container is absent.

Table 1: Data Dictionary excerpt for the semantic unit size

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 has been retrieved and that an application has enough room to move or process files. It might also be used when billing for storage.		
Data constraint	Integer		
Object category	**Intellectual Entity / Representation**	**File**	**Bitstream**
Applicability	Not applicable	Applicable	Applicable
Examples		2038937	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.		

4.2. Sample Data Dictionary Entry for a Container Unit

Table 2 shows the beginning of the Data Dictionary entry for objectCharacteristics, the container unit for size. You can tell it is a container because it has semantic components and the data constraint is "container." Note that the included semantic components can be unitary, like size, or containers themselves, like format.

Table 2: Data Dictionary excerpt for semantic unit objectCharacteristics

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 objectCharacteristicsExtension.		
Data constraint	Container		
Object category	Intellectual Entity / Representation	File	Bitstream
Applicability	Not applicable	Applicable	Applicable
Repeatability		Repeatable	Repeatable
Obligation		Mandatory	Mandatory
Usage notes	The semantic units included in *objectCharacteristics* should be treated as a set of information that pertains to a single object at a single *compositionLevel*. 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 *objectCharacteristics* would have an incrementally higher *compositionLevel*. When encryption is applied, the *objectCharacteristics* block must include an inhibitors semantic unit. 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.		

5. PREMIS IN USE

5.1. PREMIS in XML

There is an expectation (although not a requirement) that when PREMIS is used for exchange it will be represented in XML. The PREMIS Maintenance Activity provides an XML schema that corresponds directly to the Data Dictionary to provide a straightforward description of Objects, Events, Agents and Rights. Figure 5 shows a snippet of PREMIS metadata using the PREMIS XML schema.

In practice, most preservation systems already use XML formats for importing and exporting data. Many use METS (Metadata Encoding and Transmission Standard), another standard maintained by the Library of Congress, as an XML container for bringing different types of metadata together. It is possible to use PREMIS inside of METS, but this is not entirely straightforward for two reasons. First, METS breaks up information into different sections according to whether it is technical metadata, rights metadata, or provenance metadata. The PREMIS schema, following the Data Dictionary, has sections for Objects, Rights, Events and Agents. There is some correspondence between the two structures but it isn't perfect, especially for Agent information. Second, PREMIS and METS have some overlap; for example, each defines a tag for storing checksums. If the two are used together, you have to decide whether to record these overlapping elements in PREMIS sections, METS sections, or both.

Obviously, if every implementation were to make its own decisions there could be great variation in how the data are represented, impeding interoperability. Therefore, implementers have developed best practices for using PREMIS and METS together. Guidelines for Using PREMIS with METS for Exchange are available on the PREMIS Maintenance Activity website at:
http://www.loc.gov/standards/premis/guidelines2017-premismets.pdf

```
<event>
     <eventIdentifier>
          <eventIdentifierType>DAITSS</eventIdentifierType>
          <eventIdentifierValue>10012</eventIdentifierValue>
     </eventIdentifier>
     <eventType>Format Validation</eventType>
     <eventDateTime>2008-05-06T10:40:22-04:00</eventDateTime>
     <eventOutcomeInformation>
          <eventOutcome>Invalid</eventOutcome>
          <eventOutcomeDetail>
                   <eventOutcomeDetailNote>ill-formed DateTime value<eventOutcomeDetailNote>
          </eventOutcomeDetail>
     </eventOutcomeInformation>
</event>
```

Figure 4: A snippet of PREMIS in XML

5.2. PREMIS conformance

The PREMIS specification contains a section on what it means for a repository to be PREMIS-conformant. The conformance statement, which the PREMIS Editorial Committee first issued in October 2010 and revised in 2015, specifies principles of use, levels of conformance, and guidance on implementing. It is available at: http://www.loc.gov/standards/premis/premis-conformance-20150429.pdf.

Principles of use

1) If the repository implements (stores or exports) a data element that purports to be a PREMIS semantic unit, the data element should have the same definition, data constraints and applicability as the semantic unit defined in PREMIS. If a metadata element shares the definition of a PREMIS semantic unit but does not share its name, the repository must establish a mapping between the metadata element and its corresponding PREMIS semantic unit.

2) If the repository implements a PREMIS semantic unit, its repeatability and obligation can be more stringent but not more liberal than PREMIS requires. That is, a repeatable semantic unit can be implemented as non-repeatable but not vice versa, and a mandatory element cannot be made optional.

3) An implementation must include the mandatory semantic units for any Data Model Entity (Object, Event, Agent or Rights) supported by the repository. Note that a mandatory semantic component is only included if the parent container is implemented.

4) If the repository exports information for use by another repository, it must supply values for all of the semantic units that are mandatory in the Data Dictionary. There is some flexibility in this, however, because repositories are not required to support mandatory semantic units for types of Entities that they do not support. In other words, a repository is free to support or not support PREMIS Agent, but if it does support the use of Agent, then agentIdentifier is mandatory. Similarly, a particular repository may not support Bitstream Objects, in which case it does not have to provide the otherwise mandatory Bitstream identifier.

Levels of conformance

The levels specify three ways of asserting conformance with PREMIS in a repository system:

- ***Level 1.*** Being able to map preservation metadata to PREMIS,
- ***Level 2.*** Being able to export preservation metadata as PREMIS, and
- ***Level 3.*** Using PREMIS as an internal schema in a way that does not require any further mapping or conversion.

These levels are further divided into categories: implementation of the Object Entity only or implementation of the Object Entity plus one or more other Entities. In other words, a repository must at least implement information about the Object, but is not required to support all of the Entity types defined in the PREMIS data model. It is also not required to store metadata internally using the names of PREMIS semantic units, or using values that follow PREMIS data constraints. In other words, it does not matter how a repository "knows" a PREMIS value – by storing it with the same name or different name, by mapping from another value, by pointing to a registry, by inference, by default, or by any other means. So long as the repository can provide a good PREMIS value when required, it is conformant.

On the other hand, the more semantic units a repository supports, the more value it gets from using PREMIS. The PREMIS Data Dictionary was developed to identify the "core" information most repositories will need in order to preserve digital content over the long term. A responsible preservation repository should look carefully at PREMIS and have a good reason for failing to implement any part of the Data Dictionary.

6. FOR MORE INFORMATION

The PREMIS Maintenance Activity Website (www.loc.gov/standards/premis/) has something for everyone, including links to the PREMIS Implementers Group (PIG), PREMIS implementation fairs and tutorials, schemas, tools, and news. It also has a section "PREMIS Resources" that links to literature about PREMIS and related topics (www.loc.gov/standards/premis/bibliography.html). Some of the more useful resources for general readers are listed here:

On preservation metadata in general:

> "Preservation Metadata" 2nd edition *(PDF:986KB/36pp.)*
> Brian Lavoie (OCLC) and Richard Gartner (Oxford)
> Published by the Digital Preservation Coalition as DPC Technology Watch Report No. 13-03: May 2013.
> dx.doi.org/10.7207/twr13-03

On implementing PREMIS:

> Digital Preservation Metadata for Practitioners: Implementing PREMIS.
> Angela Dappert, Rebecca Squire Guenther, Sébastien Peyrard, Editors. Springer, 2016.
> dx.doi.org/10.1007/978-3-319-43763-7

On changes in version 3.0 (Webinar):

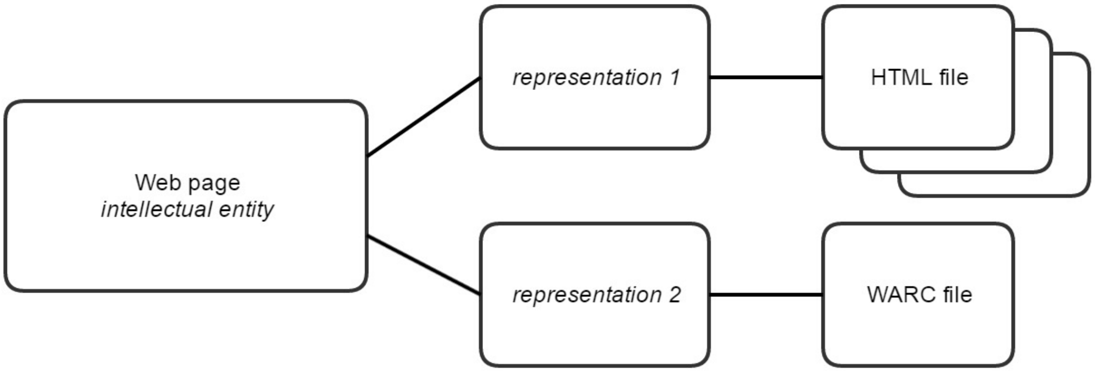
> "Digital Preservation Metadata and Improvements to PREMIS in Version 3.0: A DCMI/ASIST Joint Webinar Presented by Angela Dappert" (Wednesday, May 27, 2015)
> www.loc.gov/standards/premis/v3/tutorial.html

Appendix A: Examples

A.1. Object Example

This example describes how a website could be modelled in a hypothetical preservation system. The example walks through a high-level diagram, some details on how semantic units can be populated for the example, and finally, some snippets of XML to show how the semantic units could be implemented in the repository. The examples do not show all the semantic units that are relevant for each level of the object.

In this example, the website is called the "PREMIS Website." This is the Intellectual Entity that is to be preserved. There are two Representations of this website. For the sake of this example, one is a preservation master, the other an access master.



Intellectual Entity

The Intellectual Entity level in this example sets up one significant property of the website. This governs the behavior of both Representations. It is of course far more likely that any significant properties would be far more granular, but the example serves to show how they can be used at the IE level.

PREMIS Semantic Unit	**IE1**
1.1 objectIdentifier	
1.1.1 objectIdentifierType	IEPID
1.1.1 objectIdentifierValue	17415491
1.4 significantProperties	
1.4.1 significantPropertiesType	behavior
1.4.2 significantPropertiesValue	External links open in new browser window

Representation

The preservation level describes the institutional policy for this Object. The preservation level may be set at the IE, Representation or File level. In this case it has been set at the Representation level as the institution wishes each Representation to be cared for differently. The institution in this example has decided that the Representation comprised of the WARC file (Representation 2) is the one that they consider the preservation master; the one that will be retained in perpetuity and from which access masters are made. Representation 1 is the access master with the purpose of being the Representation that is used by the public to view. Its value is not long-term, it is to be retained for as long as the current access mechanism is in place.

Representation 1 has been assigned a “bit preservation” value. The institutional policy defines this as requiring only backup copies are made and no risk analysis or resulting emulation or migration is required.

Representation 2 on the other hand has been classed as “logical preservation” with full capability. This is defined by the institution as the Representation undergoing the relevant risk analysis, and where required, emulation or migration is used to ensure preservation into the future.

PREMIS Semantic Unit	**Representation 1**	**Representation 2**
1.1 objectIdentifier		
1.1.1 objectIdentifierType	REPPID	REPPID
1.1.1 objectIdentifierValue	17415492	17415495
1.3 preservationLevel		
1.3.1 preservationLevelType	bit preservation	logical preservation
1.3.2 preservationLevelValue	low	full
1.3.3 preservationLevelRole	capability	intention
1.3.4 preservationLevelRationale	institutional policy	institutional policy
1.3.5 preservationLevelDateAssigned	2015-02-23	2015-02-23
1.13 relationship		
1.13.1 relationshipType	derivation	derivation
1.13.2 relationshipSubType	has source	Is source of
1.13.3 relatedObjectIdentifier		
1.14.3.1 relatedObjectIdentifierType	REPPID	REPPID
1.14.3.2 relatedObjectIdentifierValue	17415495	17415492

An XML snippet for the preservation level could look like:

```
<premis:preservationLevel>
    <premis:preservationLevelType>logical preservation</premis:preservationLevelType>
    <premis:preservationLevelValue>full</premis:preservationLevelValue>
    <premis:preservationLevelRole authority="preservationLevelRole"
        authorityURI="http://id.loc.gov/vocabulary/preservation/preservationLevelRole"
        valueURI="http://id.loc.gov/vocabulary/preservation/preservationLevelRole/int">intention
    </premis:preservationLevelRole>

    <premis:preservationLevelDateAssigned>2015-02-23
</premis:preservationLevel>
```

Files

Representation 1 includes a number of discrete files, each of which are described as File objects with their own set of metadata elements.

PREMIS Semantic Unit	File 1	File 2
1.1 objectIdentifier		
1.1.1 objectIdentifierType	FILEPID	FILEPID
1.1.1 objectIdentifierValue	17415493	17415494
1.5 objectCharacteristics		
1.5.1 compositionLevel	0	0
1.5.2 fixity		
1.5.2.1 messageDigestAlgorithm	SHA256	SHA256
1.5.2.2 messageDigest	d2bed92b73c7090bb30a0b30016882e7069c437488e1513e9deaacbe29d38d92	074862dbfa0806ef5a26c3ff748d394e79728e9c957ff8c198ae13214c9cfec0
1.5.2.3 messageDigestOriginator	NRI	NRI
1.5.3 size	4859385	123451
1.5.4 format		
1.5.4.1 formatDesignation		
1.5.4.1.1 formatName	Extensible Hypertext Markup Language	JPEG File Interchange Format
1.5.4.1.2 formatVersion	1	1.02
1.5.4.2 formatRegistry		
1.5.4.2.1 formatRegistryName	PRONOM	PRONOM
1.5.4.2.2 formatRegistryKey	fmt/102	fmt/44
1.5.4.2.3 formatRegistryRole	specification	specification
1.13 relationship		
1.13.1 relationshipType	structural	structural
1.13.2 relationshipSubType	Is included in	Is included in
1.13.3 relatedObjectIdentifier		
1.14.3.1 relatedObjectIdentifierType	REPPID	REPID
1.14.3.2 relatedObjectIdentifierValue	17415492	17415492

An XML snippet for objectCharacteristics for object 1would look like:

```
 <premis:objectCharacteristics>
    <premis:compositionLevel>0</premis:compositionLevel>
    <premis:fixity>
       <premis:messageDigestAlgorithm>SHA256</premis:messageDigestAlgorithm>
       <premis:messageDigest>d2bed92b73c7090bb30a0b30016882e7069c437488e1513e9deaacbe29d38d92
           </premis:messageDigest>
       <premis:messageDigestOriginator> NRI</premis:messageDigestOriginator>
    </premis:fixity>
    <premis:size>4859385</premis:size>
    <premis:format>
       <premis:formatDesignation>
           <premis:formatName>Extensible Hypertext Markup Language</premis:formatName>
           <premis:formatVersion>1</premis:formatVersion>
           <premis:formatRegistry>
               <premis:formatRegistryName>PRONOM</premis:formatRegistryName>
               <premis:formatRegistryKey>fmt/102</premis:formatRegistryKey>
               <premis:formatRegistryRole>specification</premis:formatRegistryRole>
           </premis:formatRegistry>
    </premis:format>
</premis:objectCharacteristics>
```

A.2. Event Example

This example describes an event undertaken by the repository on an object. A common function of a repository is to identify the file format of an object and to validate that it complies with standards for that format, noting compliance or exceptions. The example illustrates format validation of File 1 in the previous Object example.

PREMIS Semantic Unit	File 1
2.1 eventIdentifier	
2.1.1 eventIdentifierType	DPS
2.1.1 eventIdentifierValue	25
2.2 eventType	validation
2.3 eventDateTime	2013-07-01T17:23:25Z
2.4 eventDetailInformation	
2.4.1 eventDetail	Format identification performed on file
2.5 eventOutcomeInformation	
2.5.1 eventOutcome	success
2.5.2 eventOutcomeDetail	
2.5.2.1 eventOutcomeDetailNote	FORMAT_ID=fmt/102;IDENTIFICATION_METHOD=SIGNATURE;FILE_EXTENSION=html;DEPOSIT_ACTIVITY_ID=659202;PID=FL26274026;SIP_ID=444381;PRODUCER_ID=29408010705;TASK_ID=48;PROCESS_ID=;MF_ID=8
2.6 linkingAgentIdentifier	
2.6.1 linkingAgentIdentifierType	local
2.6.2 linkingAgentIdentifierValue	DROID6_63_1
2.6.3 linkingAgentIdentifierRole	implementer
2.7 linkingObjectIdentifier	
2.7.1 linkingObjectIdentifierType	FILEPID
2.7.2 linkingObjectIdentifierValue	17415493

An XML snippet for the Event of format identification could look like:

```
<premis:event>
    <premis:eventIdentifier>
        <premis:eventIdentifierType>DPS</premis:eventIdentifierType>
        <premis:eventIdentifierValue>25</premis:eventIdentifierValue>
    </premis:eventIdentifier>
    <premis:eventType authority=”event type”
                      authorityURI=”http://id.loc.gov/vocabulary/preservation/eventType”
                      valueURI=”http://id.loc.gov/vocabulary/preservation/eventType/val”>
                      validation</premis:eventType>
    <premis:eventDateTime>2013-07-01T17:23:25Z</premis:eventDateTime>
    <premis:eventDetailInformation>
        <premis:eventDetail>Format identification performed on file</premis:eventDetail>
    </premis:eventDetailInformation>
    <premis:eventOutcomeInformation>
        <premis:eventOutcome>success</premis:eventOutcome>
        <premis:eventOutcomeDetail>
        <premis:eventOutcomeDetailNote>
           FORMAT_ID=fmt/102;IDENTIFICATION_METHOD=SIGNATURE;FILE_EXTENSION=html;
           DEPOSIT_ACTIVITY_ID=659202;PID=FL26274026;SIP_ID=444381;
           PRODUCER_ID=29408010705;TASK_ID=48;PROCESS_ID=;MF_ID=8</premis:eventOutcomeDetailNote>
        </premis:eventOutcomeDetail>
    </premis:eventOutcomeInformation>
    <premis:linkingAgentIdentifier>
        <premis:linkingAgentIdentifierType>LOCAL</premis:linkingAgentIdentifierType>
        <premis:linkingAgentIdentifierValue>DROID6_63_1</premis:linkingAgentIdentifierValue
        <premis:linkingAgentRole authority="eventRelatedAgentRole"
           authorityURI="http://id.loc.gov/vocabulary/preservation/eventRelatedAgentRole"
           valueURI="http://id.loc.gov/vocabulary/preservation/eventRelatedAgentRole/imp">
           implementer</premis:linkingAgentRole>
    </premis:linkingAgentIdentifier>
    <premis:linkingObjectIdentifier>
        <premis:linkingObjectIdentifierType>FILEPID</premis:linkingObjectIdentifierType>
        <premis:linkingObjectIdentifierValue>17415493</premis:linkingObjectIdentifierValue>
    </premis:linkingObjectIdentifier>
</premis:event>
```

A.3. Agent Example

This example describes the agent associated with the event described above. In this case the agent is software that performed the action on the object.

PREMIS Semantic Unit	
3.1 agentIdentifier	
3.1.1 agentIdentifierType	local
2.1.1 agentIdentifierValue	DROID6_63_1
3.2 agentName	REG_SA_DROID
3.3 agentType	software
3.4 agentVersion	Version 6.01
3.5 agentNote	Signature version Binary SF v.63/ Container SF v.1

An XML snippet for the Agent information could look like:

```
<premis:agent>
     <premis:agentIdentifier>
           <premis:agentIdentifierType>local</premis:agentIdentifierType>
           <premis:agentIdentifierValue>DROID6_63_1</premis:agentIdentifierValue>
     </premis:agentIdentifier>
     <premis:agentName>REG_SA_DROID</premis:agentName>
     <premis:agentType>software</premis:agentType>
     <premis:agentVersion>Version 6.01</premis:agentVersion>
     <premis:agentNote>Signature version Binary SF v.63/ Container SF    v.1</premis:agentNote>
</premis:agent>
```

Appendix B: Glossary of terms

This glossary brings together definitions that appeared earlier in the text of this guide. The definitions may be less formal that those appearing in the *PREMIS Data Dictionary for Preservation Metadata*.

Actionable: The quality of being recorded in a controlled form that can be acted upon by computer program.

Agent: A person, organization or computer program that has a role pertaining to an *Event* or a statement of *Rights*.

Bitstream Object: A type of PREMIS *Object*; data within a file that have common properties for preservation purposes and cannot stand alone.

Container units: *Semantic units* which take no value of their own but exist to group related subunits.

Digital provenance: Documentation of the chain of custody and change history of a digital resource.

Emulation: A *preservation strategy* that involves reproducing an old rendering environment on newer hardware and/or software.

Environment: The hardware, software and other objects required to render an Object.

Environment Object: An Object that is part of the technical stack of software, hardware and other dependencies needed to correctly interpret the Representations, Files and Bitstreams.

Event Entity: A PREMIS *Entity* that aggregates information about actions that affect *Objects* in the repository.

Extension container: A special type of PREMIS *container unit* that has no subunits defined under it but is designed as a placeholder for non-PREMIS metadata.

File Object: A type of PREMIS *Object*; a computer file, like a PDF or JPEG.

Inhibitors: Features of a digital object intended to restrict access, use or *migration*.

Intellectual Entity: A set of content that is treated as a unit for purposes of management and description; similar to a "bibliographic entity" in library science.

Migration: A *preservation strategy* that involves making a version of a digital file in a newer file format.

Objects: Digital items that are actually stored and managed in a preservation repository. PREMIS defines four types of Objects: *Files*, *Bitstreams, Representations* and *Intellectual Entities*.

Preservation metadata: Metadata that support activities intended to ensure the long-term usability of a digital resource.

Preservation strategies: Techniques employed to ensure that digital resources remain usable over the long term; two common strategies are *migration* and *emulation*.

Representation Object: A type of PREMIS *Object*; the set of all *File Objects* needed to render an *Intellectual Entity*.

Rights entity: A PREMIS *Entity* that aggregates information about rights and permissions pertaining to Objects in a preservation repository.

Semantic units: Pieces of information or knowledge.

Significant properties: Characteristics of an Object that should be maintained through preservation actions.