Library of Congress Network Development and MARC Standards Office

Understanding PREMIS

Authored By: Priscilla Caplan

Date: February 1, 2009

UNDERSTANDING PREMIS

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 master the *PREMIS Data Dictionary for Preservation Metadata*, this guide may serve as a gentle introduction that makes the larger document feel more familiar.

In this guide, terms in the `Verdana font` are names of PREMIS semantic units. Terms that appear in the Glossary are *italicized* on their first reference.

1. PREMIS IN CONTEXT

1.1. What is preservation metadata?

If you work in a library or archives, 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 displayed or otherwise used. 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 transforming old formats to newer equivalents (*migration*), or imitating the old rendering environment on newer hardware and software (*emulation*). Both migration and emulation strategies require metadata about the original file formats and the hardware and software environments supporting them.
- Preservation actions may entail changing original resources or changing how they are rendered. 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. An updated second version was issued in March 2008. The Library of Congress maintains a schema for representing PREMIS in XML.

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 in using PREMIS, and commissions studies and publications related to PREMIS, like this guide.

Usually, when people refer to "PREMIS" they 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/v2/premis-2-0.pdf
PREMIS Website: www.loc.gov/standards/premis/
PREMIS Implementers Group discussion list: listserv.loc.gov/listarch/pig.html

1.3. What is in the PREMIS Data Dictionary?

The PREMIS Data Dictionary defines a core set of *semantic units* (see section 2.1) 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, such as MARC21, MODS, and Dublin Core.
- Detailed information about media or hardware. Again, although clearly relevant to preservation, this metadata was left to other communities to define.

- Information about agents (people, organizations or software) other than the minimum needed for identification.
- Information about rights and permissions, except those that directly 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.

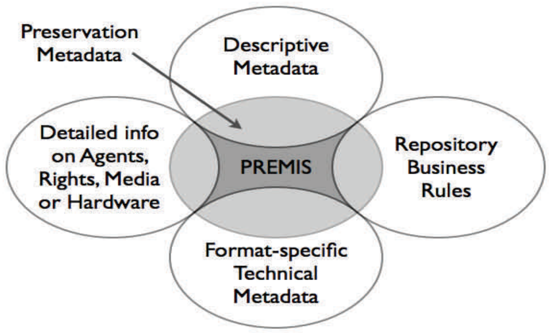

Preservation Metadata
Descriptive Metadata
Detailed info on Agents, Rights, Media or Hardware
PREMIS
Repository Business Rules
Format-specific Technical Metadata

Figure 1. PREMIS as a subset of all Preservation Metadata

Figure 1 shows all metadata relevant to preservation as the shaded circle in the center of the diagram. The circle 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, shown here in darker gray, 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 among preservation repositories.

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, or a customer may want to switch from one third-party service to another. 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 taking a PREMIS tutorial if you can; these are periodically offered by the PREMIS Maintenance Activity.

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 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. It is difficult to describe inhibitors by program, because the program may be prevented from analyzing the object, so if you know a file has inhibitors, it is important to note this. 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. However, the PREMIS event types are mostly designed to describe actions that happen after something is submitted for ingest to a repository. To record events that happen before ingest, such as acquisition and accession, you may need to invent your own event types.

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. There are a number of initiatives dedicated to better modeling and description of significant properties, but these are still in their early stages. 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 for future use.

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. You should be careful to record any known rights information, including copyright status, license terms and special permissions.

2. DATA DICTIONARY CONVENTIONS

2.1. Semantic Units

The PREMIS Data Dictionary defines semantic units, not metadata elements. The distinction is subtle but important. 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 rather abstract semantic units. PREMIS semantic units have a direct mapping to the metadata elements defined in the PREMIS XML schema, and may have a less direct mapping to metadata in other schema.

The names of PREMIS semantic units are "camel case" strings. That is, words are not separated by spaces but by capital letters: 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 in version 2.0 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 semantic unit objectIdentifier is mandatory (M) and repeatable (R). Because there are semantic units defined underneath it, you can deduce that objectIdentifier does not hold a value itself but serves as a container for the component elements objectIdentifierType and objectIdentifierValue. Because objectIdentifierType and objectIdentifierValue 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, but is very important information for digital preservation. The extension container objectCharacteristicsExtension gives a place to record technical metadata defined by other data dictionaries, such as the Z39.87 standard for describing bitmap images.

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.

3. PREMIS DATA MODEL

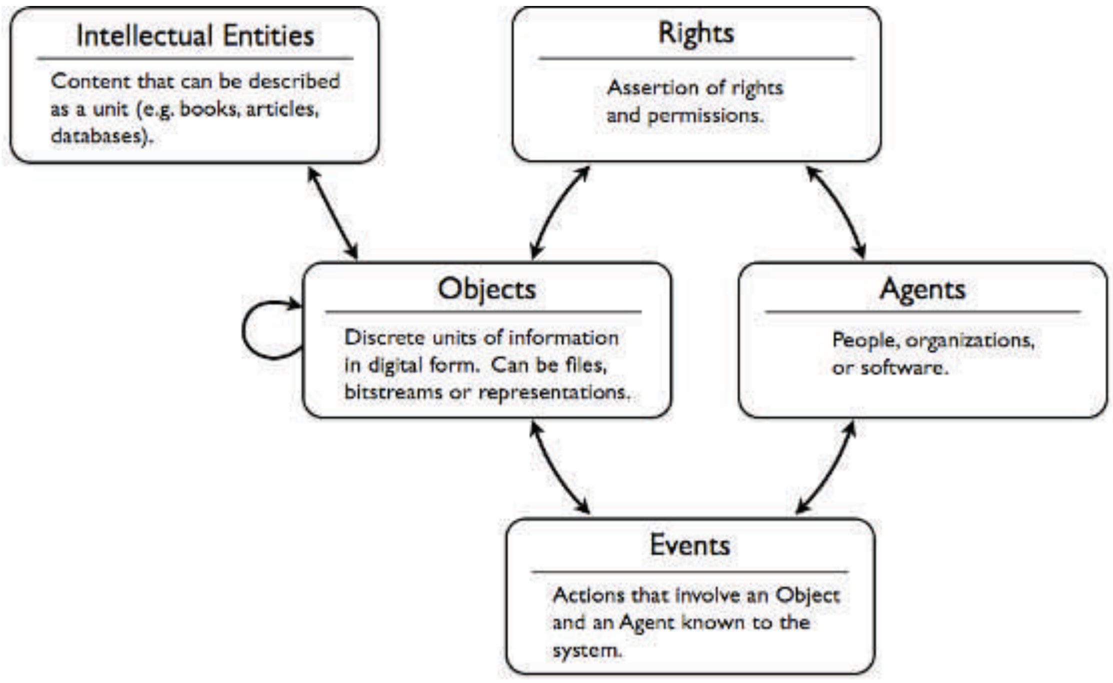

Intellectual Entities
Content that can be described as a unit (e.g. books, articles, databases).
Rights
Assertion of rights and permissions.
Objects
Discrete units of information in digital form. Can be files, bitstreams or representations.
Agents
People, organizations, or software.
Events
Actions that involve an Object and an Agent known to the system.

Figure 2. PREMIS Data Model

One of the main principles behind PREMIS is that you need to be very clear about what you are describing. PREMIS defines five kinds of things (called *entities*) you can talk about: Intellectual Entities, Objects, Agents, Events and Rights. This is called the PREMIS data model, shown in Figure 2 above.

3.1. Intellectual Entity

Intellectual Entities are conceptual, and might be called "bibliographic entities." PREMIS defines an Intellectual Entity 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 actually define any metadata pertaining to Intellectual Entities because there are plenty of descriptive metadata standards to choose from.

PREMIS does say an object in a preservation system should be associated with the 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 in the PREMIS description of the ebook.

3.2. 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.

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. 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. Simple environment information, such as a single environment that is known to work, can be easily recorded in PREMIS, but comprehensive environment information can be difficult and time-consuming to determine. When detailed information applies to classes of objects rather than local instances of objects, as is the case with environment information, it can be usefully aggregated in registries and shared by all repositories. PREMIS allows repositories to link to information stored in external registries when this is preferable to storing it locally.

PREMIS actually defines three different kinds of objects and requires implementers to make a distinction between them. These are *file* objects, *representation* objects, and *bitstream* objects.

A file object is just what it sounds like: a computer file, like a PDF or JPEG file.

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 any characteristics of the totality (e.g. the Web page as a whole) that may be different from those of its parts.

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.

Some semantic units defined in the PREMIS data dictionary are applicable to all three types of object, while others are applicable to only one or two types of object. Having different types of object forces you to really think about what you are describing and be as precise as possible, which is important for machine-processing.

3.3. 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 the Data Dictionary entry for eventType provides a "starter list" of important event types to encourage repositories to record these events consistently.

3.4. Agents

Agents are actors that have roles in events and in rights statements (see 3.5 Rights). Agents can be people, organizations, or software applications. PREMIS defines only a minimum number of semantic units necessary to identify agents, since there are several external standards that can be used to record more detailed information. (For a sampling of these see "Metadata standards and specifications for describing people and their interests" at www.ukoln.ac.uk/metadata/resources/people/.) 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).

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. An agent's role in relation to an event or rights statement is considered a property of the event entity or rights entity, not of the agent itself.

3.5. Rights

Most preservation strategies involve making identical copies and derivative versions of digital objects, actions that are restricted by copyright law to the rights holders. The *Rights entity* aggregates information about rights and permissions that are directly relevant to preserving objects in the repository. 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 or statute,
- 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 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 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

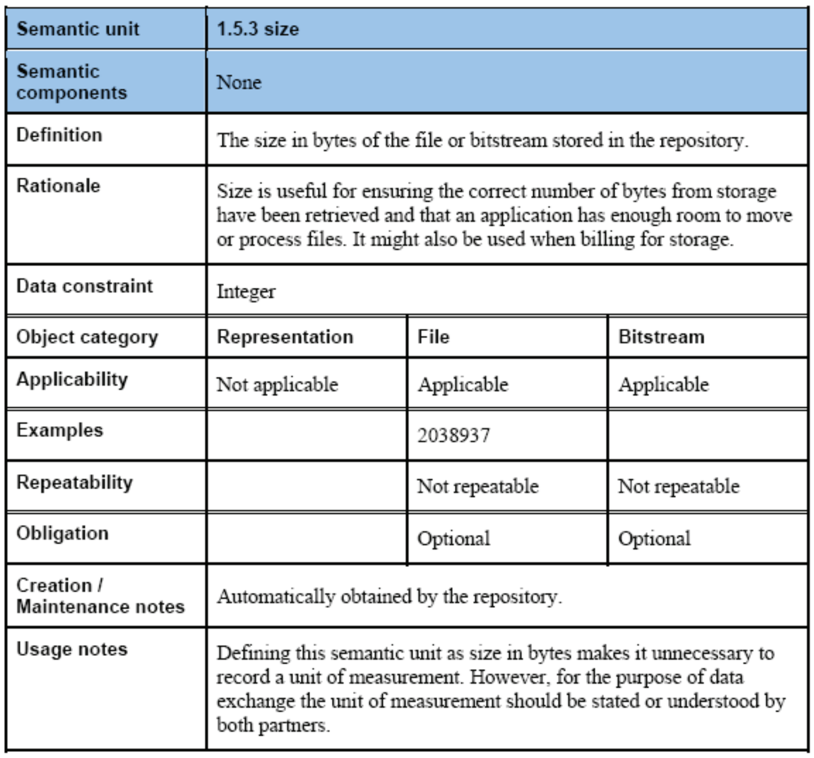
Figure 3 shows the Data Dictionary entry for the semantic unit size, which is a component or subunit of the container called objectCharacteristics. size itself has no subunits. The Data Dictionary entry includes a definition of the element and a reason (rationale) for including it among the core PREMIS 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 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. Sometimes the terms in the vocabulary are specified in the Data Dictionary and sometimes they are not, but in all cases 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 Maintenance Activity is developing an XML schema solution.

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, repository management 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.)

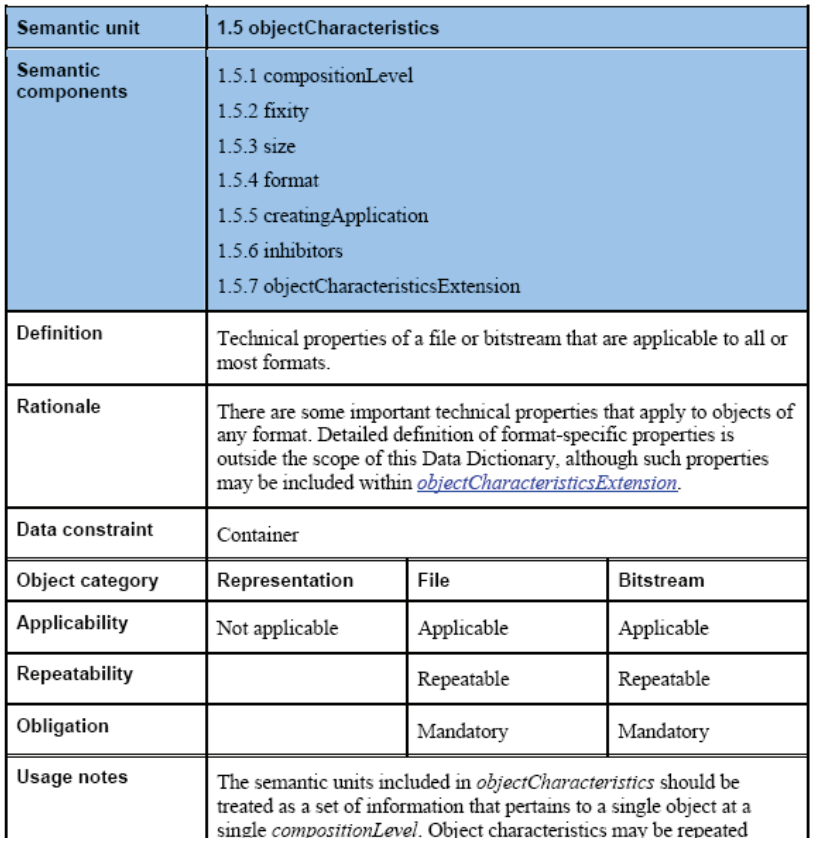


Semantic unit	1.5.3 size		
Semantic components	None		
Definition	The size in bytes of the file or bitstream stored in the repository.		
Rationale	Size is useful for ensuring the correct number of bytes from storage have been retrieved and that an application has enough room to move or process files. 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.		

Figure 3: Data Dictionary excerpt for the semantic unit size.

4.2. Sample Data Dictionary Entry for a Container Unit

Figure 4 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.



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	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		

Figure 4: Data Dictionary excerpt for semantic unit objectCharacteristics.

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 preservation repository were to make its own decisions there could be great variation in how the data is represented, impeding interoperability. Therefore there are several efforts in process to help define best practices for using PREMIS and METS together. See the PREMIS Maintenance Activity website for progress on these efforts.

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

Figure 5: A snippet of PREMIS in XML.

5.2. PREMIS conformance

The PREMIS Data Dictionary contains a section on what it means for a repository to be PREMIS-conformant. Essentially, there are three requirements:

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.

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 can not be made optional.

3) 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 Agents, but if it does support the use of Agents, 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.

These requirements should be considered in the context of a number of things that are NOT required for conformance. As noted above, a repository 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.

In the future the PREMIS maintenance activity may reconsider conformance requirements and make them more stringent, but with version 2.0 it is fairly easy for any repository to be PREMIS 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 tutorials, schemas, tools, and news. It also has a section "Resources: Articles and Presentations" that links to literature about PREMIS and related topics. Some of the more useful resources for general readers are listed here:

On preservation metadata in general:
"Preservation Metadata" *(PDF:209K/21pp.)*
Brian Lavoie (OCLC) and Richard Gartner (Oxford)
Joint report of OCLC, Oxford Library Services, and the Digital Preservation Coalition. Published as DPC Technology Watch Report No. 05-01: September 2005. http://www.dpconline.org/docs/reports/dpctw05-01.pdf

On the difference between PREMIS version 1.0 and 2.0:
"PREMIS with a Fresh Coat of Paint: Highlights from the Revision of the PREMIS Data Dictionary for Preservation Metadata "
Brian Lavoie, *D-Lib Magazine*, May/June 2008.
http://www.dlib.org/dlib/may08/lavoie/05lavoie.html

On using PREMIS and METS together:
"Battle of the Buzzwords: Flexibility vs. Interoperability When Implementing PREMIS with METS"
Rebecca Guenther, *D-Lib Magazine*, July/August 2008.
http://www.dlib.org/dlib/july08/guenther/07guenther.html

Appendix A: List of all PREMIS Semantic Units

From the *PREMIS Data Dictionary for Preservation Metadata* version 2.0.

How to read this list

1.2 objectCategory (M, NR)

The semantic unit objectCategory is Mandatory and Not Repeatable. It applies to all types of objects (file, representation, bitstream).

1.3 preservationLevel (O, R) [representation, file]
1.3.1 preservationLevelValue (M, NR) [representation, file]

The semantic unit preservationLevel is Optional and Repeatable. It can be used for representation objects and file objects only. It is a container unit because there is at least one semantic unit subordinate to it, preservationLevelValue, which also applies to representations and files only.

Object 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]
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)

Event 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)

Agent Entity Semantic Units

3.1 agentIdentifier (R, M)
- 3.1.1 agentIdentifierType
- 3.1.2 agentIdentifierValue

3.2 agentName (O, R)
3.3 agentType (O, NR)

Rights 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)

- 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)

Appendix B: Object Example

This example is a simplified version of one prepared by the Library of Congress for a PREMIS tutorial. It shows the PREMIS semantic units and values used to describe a TIFF image.

LC uses handles (identifiers created and managed by an application called The Handle System) for file objects stored within the repository (see 1.1 objectIdentifier and 1.10.3 relatedObjectIdentification). The object being described is known inside the repository by its unique identifier (1.1 objectIdentifier) but its original name before ingest was "001h.tif" (1.6 originalName). We know that it is a file because of the object category (1.2 objectCategory).

The file has an MD5 checksum computed by LC's local digital content managment system (1.5.2 fixity). Its format information is given in two ways, as a MIME type and version (1.5.4.1 formatDesignation) and by pointing to the PRONOM format registry which holds a copy of the TIFF specification.

LC considers there to be two creating applications. Something (probably an unnamed RAW file) was created on a scanner using the ScandAll application and turned into a TIFF using Adobe Photoshop (1.5.5 creatingApplication). There is no encryption or other form of inhibitor, which LC felt was worth recording explicitly (1.5.6 inhibitors).

The file is stored on disk at a location identified by the directory path (1.7.1 contentLocation); the disk device is identified very specifically, presumably so the repository managers know when the data will require migration to a more current device (1.7.2 storageMedium).

LC has taken some care to record a recommended environment for rendering (displaying) the file: using Adobe Acrobat version 5.0 on an Intel x86 computer running Windows XP. The processor speed must be at least 60 MHz and there should be at least 32 MB of memory, with 64 MB recommended (1.8 environment).

The TIFF file is part of a representation with the internally assigned identifier *R200802948*. This information is given in the third block of relationship information (1.10 relationship). There are two other files in this representation, identified by handles. We know they must be file objects because the type of relationship is described as "sibling". We would have to look at the object description for *loc.music/gottlieb.09602* and *loc.music/gottlieb.mets* for more information about them, such as what types of files they are.

We do not have any descriptive information about the intellectual entity represented by this representation. However, we do have a link to its identifier, an LCCN (1.12 linkingIntellectualEntityIdentifier). Presumably, if we looked up that LCCN, we would find a bibliographic description of the intellectual entity (although not in this example, because the LCCN is made up.)

Semantic unit	**Value**
1.1 objectIdentifier	
1.1.1 objectIdentifierType	hdl
1.1.2 objectIdentifierValue	loc.music/gottlieb.09601
1.2 objectCategory	file
1.3 preservationLevel	
1.3.1 preservationLevelValue	full
1.5 objectCharacteristics	
1.5.1 compositionLevel	0
1.5.2 fixity	
1.5.2.1 messageDigestAlgorithm	MD5
1.5.2.2 messageDigest	36b03197ad066cd719906c55eb68ab8d
1.5.2.3 messageDigestOriginator	localDCMS
1.5.3 size	20800896
1.5.4 format	
1.5.4.1 formatDesignation	
1.5.4.1.1 formatName	image/tiff
1.5.4.1.2 formatVersion	6.0
1.5.4.2 formatRegistry	
1.5.4.2.1 formatRegistryName	PRONOM
1.5.4.2.2 formatRegistryKey	fmt/10
1.5.4.2.3 formatRegistryRole	specification
1.5.5 creatingApplication	
1.5.5.1 creatingApplicationName	ScandAll 21
1.5.5.2 creatingApplicationVersion	4.1.4
1.5.5.3 dateCreatedByApplication	1998-10-30
1.5.5 creatingApplication	
1.5.5.1 creatingApplicationName	Adobe Photoshop
1.5.5.2 creatingApplicationVersion	CS2
1.5.5.3 dateCreatedByApplication	1998-10-30T08:29:02
1.5.6 inhibitors	
1.5.6.1 inhibitorType	none

1.6 originalName	001h.tif
1.7 storage	
1.7.1 contentLocation	
1.7.1.1 contentLocationType	filepath
1.7.1.2 contentLocationValue	amserver/
1.7.2 storageMedium	IBM DS4000 System 1740-5208
1.8 environment	
1.8.1 environmentCharacteristic	recommended
1.8.2 environmentPurpose	render
1.8.2 environmentPurpose	edit
1.8.5 software	
1.8.5.1 swName	Adobe Acrobat
1.8.5.2 swVersion	5.0
1.8.5.3 swType	renderer
1.8.5 software	
1.8.5.1 swName	Windows
1.8.5.2 swVersion	XP
1.8.5.3 swType	OperatingSystem
1.8.6 hardware	
1.8.6.1 hwName	Intel x86
1.8.6.2 hwType	processor
1.8.6.3 hwOtherInformation	60 MHz minimum
1.8.6 hardware	
1.8.6.1 hwName	64MB RAM
1.8.6.2 hwType	memory
1.8.6.3 hwOtherInformation	32 MB minimum

1.10 relationship	
1.10.1 relationshipType	structural
1.10.2 relationshipSubType	has sibling
1.10.3 relatedObjectIdentification	
1.10.3.1 relatedObjectIdentifierType	hdl
1.10.3.2 relatedObjectIdentifierValue	loc.music/gottlieb.09602
1.10.3.3 relatedObjectSequence	0
1.10 relationship	
1.10.1 relationshipType	structural
1.10.2 relationshipSubType	has sibling
1.10.3 relatedObjectIdentification	
1.10.3.1 relatedObjectIdentifierType	hdl
1.10.3.2 relatedObjectIdentifierValue	loc.music/gottlieb.mets
1.10.3.3 relatedObjectSequence	0
1.10 relationship	
1.10.1 relationshipType	structural
1.10.2 relationshipSubType	is included in
1.10.3 relatedObjectIdentification	
1.10.3.1 relatedObjectIdentifierType	LocalRepository
1.10.3.2 relatedObjectIdentifierValue	R200802948
1.10.3.3 relatedObjectSequence	0
1.12 linkingIntellectualEntityIdentifier	
1.12.1 linkingIntellectualEntityIdentifierType	LCCN
1.12.2 linkingIntellectualEntityIdentifer Value	2007-86121

Appendix C: 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.

entity: In PREMIS, a type of thing you can talk about. PREMIS entity types are *Intellectual Entities*, *Objects*, *Agents*, *Events* and *Rights*.

environment: The hardware, software and other objects required to render an object.

event entity: A PREMIS *entity* that aggregates infomration 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 three types of objects: *files*, *bitstreams* and *representations*.

preservation metadata: Metadata that supports 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: A type of PREMIS *object*; the set of all *file objects* needed to render an i*ntellectual 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.