Using METS, PREMIS and MODS for Archiving eJournals

Angela Dappert, Markus Enders

The British Library
Boston Spa, Wetherby, West Yorkshire LS23 7BQ, UK
St Pancras, 96 Euston Road, London NW1 2DB, UK
angela.dappert@bl.uk, markus.enders@bl.uk

Abstract

As institutions turn towards developing archival digital repositories, many decisions on the use of metadata have to be made. In addition to deciding on the more traditional descriptive and administrative metadata, particular care needs to be given to the choice of structural and preservation metadata, as well as to integrating the various metadata components. This paper reports on the use of METS structural, PREMIS preservation and MODS descriptive metadata for the British Library's eJournal system.

Introduction

At the British Library, a system for ingest, storage, and preservation of digital content is being developed under the Digital Library System Programme, with eJournals as the first content stream. This was the driver for developing a common format for the eJournal Archival Information Package (AIP) as defined in OAIS [CCSDS 2002].

In order to understand metadata needs, it is helpful to understand the business processes and data structures. eJournals present a difficult domain for two reasons. The first is that eJournals are structurally complex. For each journal title, new issues are released in intervals. They may contain varying numbers of articles and other publishing matter. Articles are submitted in a variety of formats, which might vary from article to article within a single issue.

The second reason is that, the production of eJournals is outside the control of the digital repository and is done without the benefit of standards for the structure of submission packages, file formats, metadata formats and vocabulary, publishing schedules, errata, etc.. As a consequence, systems that handle eJournals need to accommodate a great variety of processes and formats. This paper presents a solution that can accommodate the complexity and variety found in eJournals.

Fortunately, there has been a substantial amount of work over recent years to define metadata specifications that can support complex cases such as eJournals. The Metadata Encoding and Transmission Specification (METS) provides a robust and flexible way to define digital objects ([METS 2006]). The Metadata Object Description Scheme (MODS) provides ways to describe objects, and builds on the library community's MARC tradition ([MODS 2006]). Finally, the Preservation Metadata Implementation Strategy (PREMIS) data dictionary ([PREMIS 2005]) provides ways of describing objects and processes that are essential for digital preservation. These three metadata specifications are all built on an XML ([XML 2006]) foundation. Their user communities and underlying approaches also have much in common. All of them are content-type independent, which makes it possible to define shared usage guidelines for the various content-types held in the archival store.

Unfortunately, there are many ways to combine these three specifications to provide a complete solution to the problem of defining an eJournal Archival Information Package. This paper explains one approach.

The eJournal Ingest Workflow

Ingesting eJournals requires a complex workflow that needs to be adjusted for each individual information provider's submission process and formats.

Each submission may contain several submission information packets (SIP) as defined in OAIS [CCSDS 2002]. Most SIPs are tarred or zipped files that need to be unpacked and virus checked before they can be processed further.

An unpacked SIP will typically contain content files, descriptive metadata for articles, issues and journals, and manifests listing the content of the SIP with size and hashing information.

Since a SIP may contain one or several issues and articles for one or several journals, each structured according to the information provider's conventions, and possibly containing special issues or supplements, the content needs to be split up into identified packages with well-defined structural relationships. The publisher supplied structural relationships between article, issues and journal objects may have been captured in the directory structure, through file naming conventions or through explicit metadata. In the latter case, issue and journal metadata may have been kept with each article's metadata, or contained as distinct metadata sets that are linked to each other. These relationships are extracted and represented in a uniform way, as specified in the British Library's METS, PREMIS and MODS application profiles.

Publisher supplied metadata may have been represented using in-house formats, standards, or modified standards. We extract metadata either from the publisher supplied metadata or directly from the content. The latter is typically the case for technical metadata. The extracted metadata is then normalized according to the British Library's METS, PREMIS and MODS application profiles.

Typically, information providers submit several manifestations of each article. A manifestation is a collection of all files that are needed to create one rendition of an arti-

cle. An HTML manifestation, for example, might consist of the HTML file and several accompanying image, video and sound files. Often the submitted content contains a marked-up representation of an article that, again, may be based on proprietary, standard, or modified standard XML schemas or DTDs.

The result of the ingest and normalization processes is one or more Archival Information Packages (AIPs) that can be stored. Structural relationships, metadata and, possibly, content are normalized in order to ensure uniform search across all digital objects and to guarantee the sustainability of formats, data and structural relationships of the AIPs.

The structure of the AIPs is tied to the technical infrastructure of the preservation system.

Technical Infrastructure

The British Library's technical infrastructure to preserve digital material consists of an ingest system, a metadata management component that may vary for different content-types, and an archival store that is shared for all content-types. They are linked with the existing integrated library system (ILS). This architecture is designed to enable access to resources, as well as to support long-term preservation activities, such as format migrations.

The eJournal *ingest system* under development is highly customizable and can be adjusted for different ingest processes, metadata formats, and ways of bundling and structuring the submitted content files. It extracts and normalizes relevant metadata and content.

The *metadata management component* (MMC) manages all types of metadata in a system-specific form, stores it in a database, and provides an interface for resource discovery and delivery. Since the ILS is designed to hold information on the journal-title and issue levels only, it is necessary to keep all article related information in the metadata management component; the system synchronizes changes to journal and issue information with the ILS.

The *archival store* is the long-term storage component that supports preservation activities. All content files are stored there. All archival metadata (that which goes beyond day-to-day administration) is linked to the content and also placed into the archival store. Even though the metadata in the archival store is not intended to be used for operational access, we consider it good archival practice to hold content and metadata within the same system to ensure that the archival store is complete within itself. This archival metadata is represented as a hierarchy of METS files with PREMIS and MODS components that reference all content files (images, full text files, etc.). The bundle of METS and content files comprises the Archival Information Package (AIP).

METS provides a flexible framework for modeling different document types and scenarios. The example of eJournal preservation will show how complex documents and their relationships are modeled in METS and stored in the system.

AIP Granularity

To understand the design of the system, it is fundamental to know that the objects in the underlying digital store are write-once in order to support archival authenticity and track the objects provenance; an in-situ update of AIPs in the digital store is not possible. Updated AIPs need to be added to the store and generations need to be managed. (A *generation* corresponds to an update to an object. Words such as *version* or *edition* are heavily over-loaded in the library community.) Updates happen for several reasons, and possibly frequently. A first possible reason is the migration of content files due to obsolete file formats. Second, errors might occur during the ingest process that will result in damaged or incomplete data. Even if effective quality assurance arrangements are made, chances are still high that potential problems are occasionally detected after data has been ingested. Third, updates to descriptive or administrative metadata that is held in the archival store may be needed. Metadata updates might happen in small (e.g., a correction of a typo) or larger scale. Fourth, even though the AIPs are designed to have no dependencies on external identifiers, it is conceivable that updates of other information systems (e.g., the ILS) might affect information stored within the AIP.

In order to deal with updates efficiently, we

- separate structural information about the relationship of the files in a manifestation from the descriptive information and from submission provenance information.
- split logically separate metadata subsets that are expected to be updated independently (journal, issue, article) into separate AIPs.

The eJournal data model, therefore, contains five separate metadata AIPs representing different kinds of objects: journals, issues, articles, manifestations, and submissions. Each one is realized as a separate METS file.

The first three are purely logical objects intended to hold relevant, mostly descriptive, metadata at that level. The remaining two are different.

A *manifestation object* is a collection of all files that are needed to create one rendition of an article. It must not be mistaken for FRBR's definition of a manifestation [IFLA 1998], but is roughly equivalent to the PREMIS representation concept. A manifestation may be original or derivative, such as presentation copies or normalized preservation copies of the article. The manifestation object holds structural information about how its files relate and provenance information about the files' origin.

A *submission object* describes one submission event, including all the tarred and zipped SIP files and a record of all activities performed during ingest. Since data can be lost or corrupted in the ingest process, or a need might arise to ingest the same datasets into a different system, we store the original data as it was provided by the publisher in the archival store linked to from its submission. In the environment of a write-once store, this granularity allows us to update data independently without creating redundant records or content files.

The set of these objects represents a hierarchical data model with well defined links from underlying entities to the direct parent. We store relationships between those AIPs in the AIPs themselves in addition to the metadata management component's database. This ensures that the archival store is a closed system that is consistent and complete within itself.

METS/MODS/PREMIS

Every AIP contains at least one XML file that uses the METS schema. METS provides a flexible framework for modeling different document types and scenarios. Using additional metadata schemas, so called extension schemas, METS can embed descriptive metadata records as well as digital provenance, rights and technical metadata. Figure 1 shows the basic sections of a METS file, which are in use in our system. We store descriptive metadata as a MODS extension to the <mets:dmdSec> section. Provenance and technical metadata are captured as PREMIS extensions to the <mets:amdSec> <mets:digiprovMD> and the <mets:amdSec> <mets:techMD> sections. If the METS file describes content files, then they are identified in the <mets:structMap> section.
Within each AIP there is only one single METS file.

```
<mets:mets TYPE="issue">

  <!-- section for descriptive metadata -->
  <mets:dmdSec>
    <mets:mdWrap MDTYPE="MODS">
      <mets:xmlData> … </mets:xmlData>
    </mets:mdWrap>
  </mets:dmdSec>

  <!-- section for administative metadata -->
  <mets:amdSec>

    <!-- section for technical metadata -->
    <mets:techMD>
      <mets:mdWrap MDTYPE="PREMIS">
      <mets:xmlData> … </mets:xmlData>
  </mets:mdWrap>
    </mets:techMD

    <!-- section for digital provenance metadata -->
    <mets:digiprovMD>
      <mets:mdWrap MDTYPE="PREMIS">
      <mets:xmlData> … </mets:xmlData>
  </mets:mdWrap>
    </mets:digiprovMD>

    <!-- section for rights metadata -->
    <mets:rightsMD>
      <mets:mdWrap MDTYPE="MODS">
      <mets:xmlData> … </mets:xmlData>
  </mets:mdWrap>
    </mets:rightsMD

  </mets:amdSec>

  <!-- section describing structural relationships -->
  <mets:structMap>
    <mets:div TYPE="issue"
          DMDID="ex01MODS01"/>
  </mets:structMap>
</mets:mets>
```

Figure 1: Embedding MODS and PREMIS in the METS container

METS / MODS / PREMIS Based Data Model

The diagrams in Figures 3, 4, and 5 describe the choices we made for representing the objects, their metadata and their relationships to each other within METS, PREMIS and MODS. As illustrated in Figure 2, METS files are represented as shaded boxes, content files are represented in white boxes. Relationships that are expressed through METS tags are shown as dashed arrows; those expressed through PREMIS tags within METS are shown as solid arrows; those expressed through MODS relationship tags within METS are shown as dotted arrows.

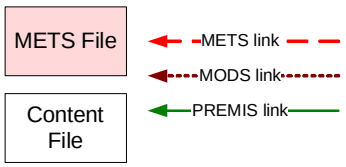

Figure 2: Legend for Figures 3, 4, and 5

Each of the five objects mentioned above is described in a separate METS file that forms a separate AIP. It stores all information about the kind of object and the relationships to other related objects.

Structural Entities: Journal, Issue and Article

Each structural entity, journal, issue and article, is stored in a separate METS file. See Figure 3 for a graphical representation.
Descriptive metadata is expressed using the MODS metadata extension schema to METS and is embedded in a single <mets:dmdSec> section of the METS file. A separate British Library MODS profile describes the elements in use and their meaning.
We use the MODS “host” link to express the hierarchical parent/child relation between journal, article, and issue.
The link uses a unique identifier that is stored within the parent object using the <mods:identifier> element. In our implementation, the identifier is called an MMC-ID (Metadata Management Component identifier).

```
<mods:mods>
  <mods:identifier type=”MMC-ID”>
    Identifier_of_object
  </mods:identifier>
  <mods:relatedItem type=”host”>
    <mods:identifier type=”MMC-ID”>
      Identifier_of_parent
    <mods:identifier>
  </mods:relatedItem>
</mods:mods>
```

Links from child to parent are suitable for systems with a write-once approach. The child objects are updated with greater frequency; for example, each new issue links to the journal. If the link was represented in the other direction, it would be necessary to create a new generation of the journal object for each issue. There are two issues that must be considered when implementing this approach. First, the identifier for the parent must be available before the child’s AIP can be defined and ingested. Second, additional indices must be used to support efficient traversal and retrieval.
As the issue or journal AIPs do not contain information about the order of the articles, and as articles may be ingested out of sequence, the position of the article within an issue must be stored within the article’s descriptive metadata. The <mods:part> element contains machine and human readable sorting information.

```
<mods:part order="w3cdtf">
  <mods:date encoding="w3cdtf">1984
  </mods:date>
```

```
<mods:detail type="volume">
  <mods:number>38</mods:number>
</mods:detail>
<mods:detail type="issue" order="1984038030">
  <mods:number>3</mods:number>
</mods:detail>
</mods:part>
```

This information can be used to create a table of contents.

The MODS <mods:relatedItem> element is also used to express a range of descriptive relationships between different objects. For example, the relationship among articles that comprise a series is expressed using the value "series" for its *type* attribute; the relationship between a journal published under a new name and the journal as it was previously known is expressed using the value "preceding" for its *type* attribute. Some of these relationships may refer to objects that are held outside the archival store using suitable persistent identifying information.

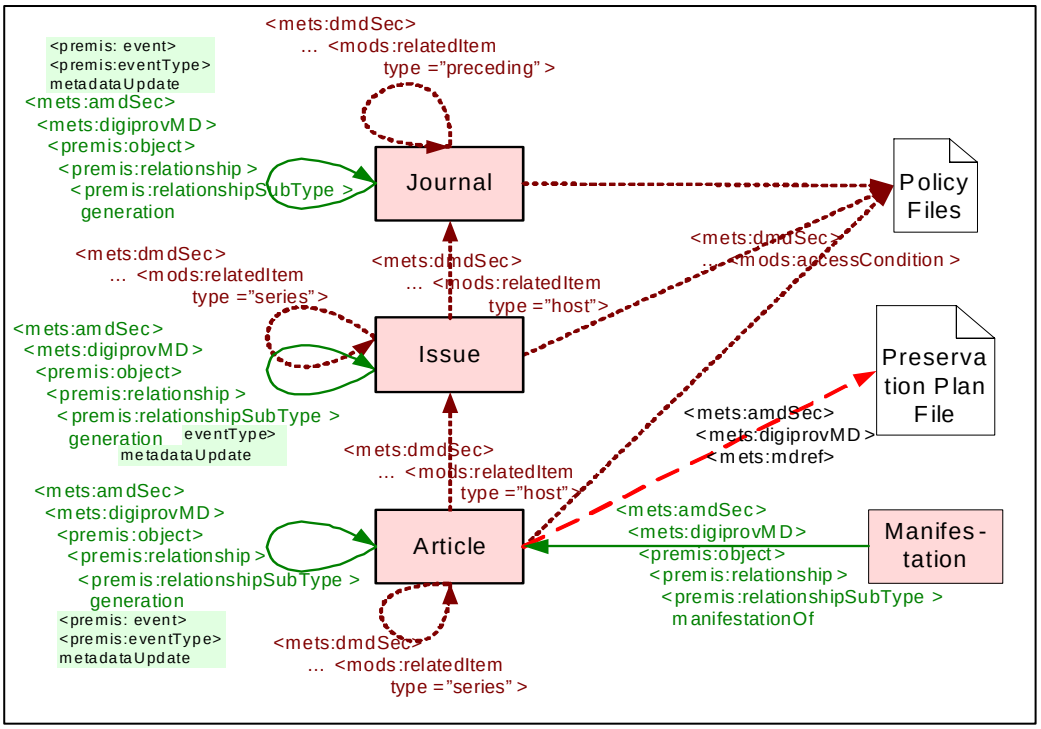

Figure 3: Data model - Structural Entities

For preservation purposes, we also store intrinsic, non-volatile rights information. This includes copyright information as well as license information. License information is stored in a separate policy AIP. Thus, every object that is licensed or acquired under a single policy can refer to the same policy object. This makes it easy to update rights information for several objects at a time without changing a large number of AIPs. The <mods:accessCondition> element is used to store the link to the policy file.

```
<mets:dmdSec> …
  <mods:mods>
    <mods:accessCondition
      type="GoverningLicense"
      xlink:href="http://xxxxx"/>
```

Similar considerations apply to Preservation Plans to which article objects link by the <mets:mdref> link.

```
<mets:amdSec>
  <mets:digiprovMD>
    <mets:mdref
      MDTYPE="OTHER"
      OTHERMDTYPE="Preservation Plan"
      LOCTYPE="OTHER"
      OTHERLOCTYPE="MMC-ID" … />
```

Provenance Metadata for Structural Entities

Long-term preservation requires us to keep a careful record of events related to digital material. Events might impact the data being preserved; data can be lost, corrupted or modified by an event. Some events won't impact the data itself, but extract information from the data to be used during its processing. Information about events is stored in the AIP's digital provenance metadata section using the PREMIS schema. Events can be associated with any object type.

As mentioned earlier, in a write-once environment, updates to metadata require the creation of a new generation of the structural entity. In this case, a new AIP for the journal, issue or article is created. This model results in several AIPs representing the different generations of one logical object. While the MODS section within the METS file defines a unique MMC-ID identifier for the logical journal, issue or article object, we need an identifier that is unique to the specific AIP of the object's generation. Journal, issue or article AIPs have a single PREMIS section under the <mets:digiProvMD> subsection that stores the AIP's identifier within the <premis:objectIdentifier> element as a generation identifier, called MMC-ID+.

```
<mets:amdSec>
  <mets:digiprovMD
    <premis:object>
      <premis:objectIdentifier>
        <premis:objectIdentifierType>
          MMC-ID +
        </premis:objectIdentifierType>
        <premis:objectIdentifierValue>
          MMC-ID.20070909:3
        </premis:objectIdentifierValue>
      </premis:objectIdentifier> …
```

The logical object can, hence, be addressed via the MMC-ID stored in MODS; the AIP that represents a

certain generation of the logical object together with information about its digital provenance can be addressed via the MMC-ID+ stored in PREMIS. This is true to our attempt of keeping logical, descriptive information in MODS, and digital provenance information in PREMIS. Each relationship in our data model is expressed via the appropriate identifier type.

The MMC-ID+ identifier is derived from the MMC-ID identifier, concatenated with a colon and a version number; it is unique for the AIP.

A <premis:relationship> "generation" link identifies the predecessor's AIP, and specifies the "metadataUpdate" event in which it had been derived from it and the <premis:agent> that executed the event.

```
<mets:amdSec>
  <mets:digiprovMD> …
    <premis:object> …
      <premis:relationship> …
        <premis:relationshipSubType>
          generation …
    <premis: event>
      <premis:eventType>metadataUpdate …
```

All digital provenance metadata is captured using the PREMIS extension schema, and is stored within the administrative metadata section of METS. Other types of provenance metadata used will be discussed below.

Manifestation

Each manifestation of an article is stored in a separate METS file. A manifestation links its actual content files together, records all events that have happened to its content files (such as uncompressing, migrating, extracting properties) and links to related versions of those files (such as the original compressed file, or the file that was the source for the migration or normalization). Manifestations are linked to their article and submission objects using the PREMIS extension schema. See Figure 4 for a graphical representation of these properties.

Each manifestation has one <mets:amdSec> that is dedicated to holding information about itself, and one for each of its files. The manifestation's <mets:amdSec> section contains the unique identifier for the manifestation and links to the article and submission objects using their MMC-ID using PREMIS

```
<mets:amdSec>
 <mets:digiprovMD> ...
  <premis:object>
   <!-- identifier of the manifestation -->
   <premis:objectIdentifier>
    <premis:objectIdentifierType>
      MMC-ID+
    </premis:objectIdentifierType>
    <premis:objectIdentifierValue>
      MMC-ID.12345:1
    </premis:objectIdentifierValue>
   </premis:objectIdentifier>
   <premis:relationship>
    <premis:relationshipType>
     Derivation
    </premis:relationshipType>
    <premis:relationshipSubType>
     manifestationOf
    </premis:relationshipSubType>
    <premis:relatedObjectIdentification>
     <premis:relatedObjectIdentifierType>
       MMC-ID
     </premis:relatedObjectIdentifierType>
     <!-- identifier of the article -->
     <premis:relatedObjectIdentifierValue>
       MMC-ID.32596:1
     </premis:relatedObjectIdentifierValue>
    </premis:relatedObjectIdentification>
   </premis:relationship>
```

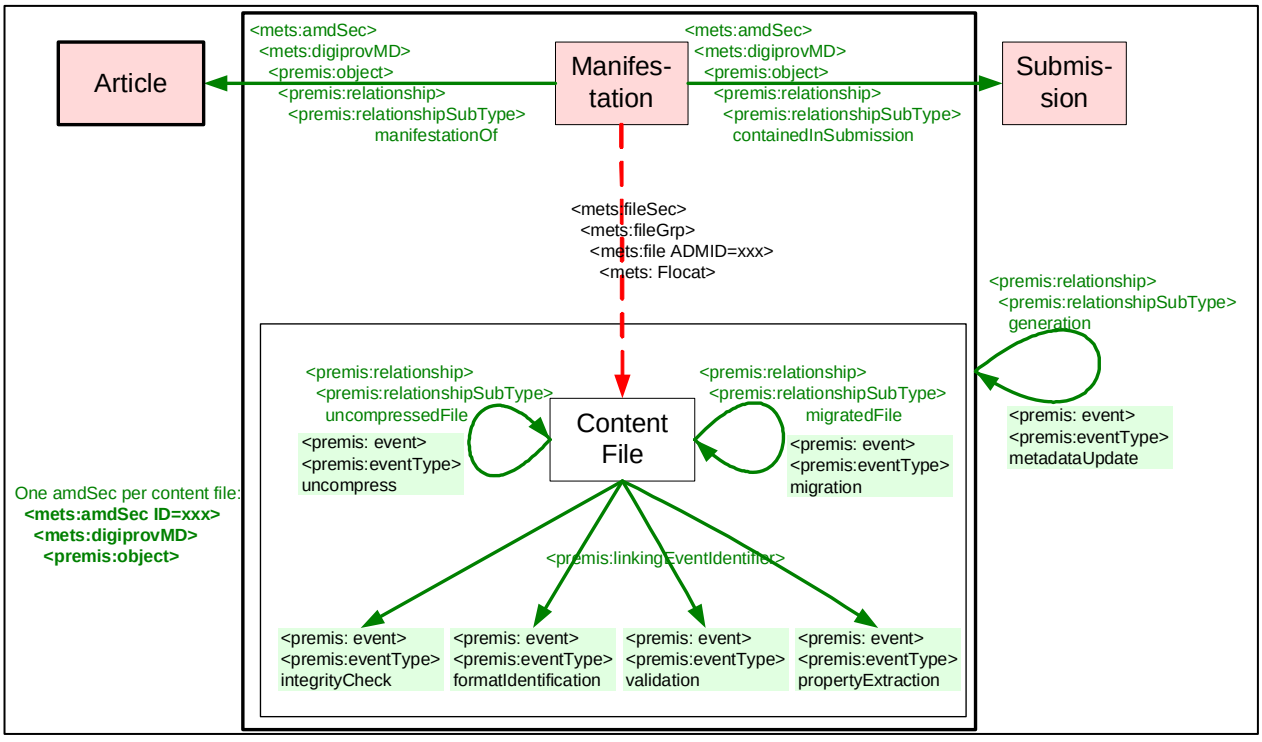

Figure 4: Data Model - Manifestation

Similarly the link to the manifestation's submission object is realized through a <premis:relationship> <premis:relationship-SubType> "containedInSubmission".
The <mets:fileSec> section is used to identify each content file of the manifestation by defining a METS ADMID administrative identifier for it, and to link to the AIPs where these files are actually stored. Each content file receives its own <mets:amdSec> section that is linked to the file by the ADMID that was defined in the <mets:fileSec>. This <mets:amdSec> section stores preservation metadata for content files.
Preservation metadata should support authenticity, understandability and identity of digital objects in a preservation context and represent the important information to preserve digital materials over a long term [PREMIS March 2008]. The METS schema does not have a section dedicated to preservation metadata. Instead it splits preservation metadata into technical, digital provenance, source and rights metadata. Therefore, preservation metadata represented in PREMIS needs to be split up and distributed over these sections. General considerations for this decision process have been discussed in [PREMIS June 2008] and [Guenther 2008].
Fixity and format information for files are regarded as technical information. Therefore the appropriate <premis:object> element containing this information is stored within the <mets:techMD> section. The digital provenance information contains basic identification and provenance information (relationships and events, as well as their attached agents) and is stored within the <mets:digiProvMD> section. As the PREMIS-container element <premis:premis> is not used, this is in accordance with the current METS-PREMIS guidelines [PREMIS June 2008].
As some of the metadata described in the PREMIS data dictionary is mandatory and the XML file won't validate without incorporating this metadata in every PREMIS section, the object-identifier as well as the object category are repeated in each section.

Provenance Metadata for Manifestations

Similar to structural entities, manifestations can have a "generation" <premis:relationship> that identifies a predecessor AIP and a "metadataUpdate" event if correction of metadata has been necessary.
In contrast, a different kind of relationship, however, is not realized as provenance metadata. When the actual semantic content of an AIP gets updated, it is usually regarded as versioning of content. Within the eJournal context a new version is created whenever the publisher decides to publish a corrected or an enhanced version of an article. From the preservation system's perspective this article is seen as a separate expression and a new AIP is created for the article as well as for its manifestation. The link between two expressions is not regarded as digital provenance metadata. For this reason the link to the previous version is stored in the descriptive MODS metadata record of the new version.

Provenance Metadata for Files

There are two different kinds of events for files: those which are side-effect free and capture information about a file, and those which result in the creation of a derivative file.
Side-effect free events include identification and validation of the file's format, extraction of properties or metadata, and validation of the file's contents to ensure its authenticity when it is disseminated, by checking and comparing the data against stored metrics, such as checksum values. These events are represented in the file's <mets:amdSec><mets:digiprovMD> section in its manifestation's METS file. Storing e.g. the metadata extraction process as an event lets us store the metadata extraction software used during this process as a related agent. As the event does not change the file, no relationship of the file to other objects is stored in the PREMIS metadata.
Derivation events produce a new bytestream while preserving its significant properties and semantic content. This will, for example, happen if an obsolete file format is regarded as "at risk" and a migration has to take place. In this case a new manifestation is created. A "migratedFile" <premis:relationship> links back from each file that results from the migration to each file that fed into the migration (One or several files can be migrated to one or several files). The "migration" event in which it had been derived and the <premis:agent> that executed the migration are recorded with the resulting file.
All files of a manifestation are referenced in its <mets:fileSec>. If a new manifestation is created during a "migration" event, in its <mets:fileSec> it may reference some unchanged files and some that resulted from the migration event. The un-affected files won't have any related file relationships.
A relationship and event similar to the "migration" event is recorded for an "uncompress" event and links from the uncompressed files to the compressed file.
Usually the event outcome should be successful. If a "migration" or an "uncompress" event fails, the file will not be ingested. However, for certain events a negative outcome will not prevent ingest. For example, if the validation of a file cannot be carried out successfully (the validation-event fails), it is handled as an exception and attempts are made to fix the file. If this is not successful, it might be decided to ingest an invalid file just to make sure that the manifestation is complete. Further attempts at fixing the file can possibly be made in the future. This event outcome is recorded within the PREMIS event.

Submission

A submission object describes a single submission event. This includes all the tarred and zipped SIP files as they have been submitted by a publisher, and a record of all activities performed during ingest. See Figure 5 for a graphical representation of these properties. The structure and relationships of the submission object are very similar to the manifestation object. Rather than content files, it references SIPs in its <mets:fileSec> and each SIP has a <mets:amdSec> of its own to hold its provenance metadata.

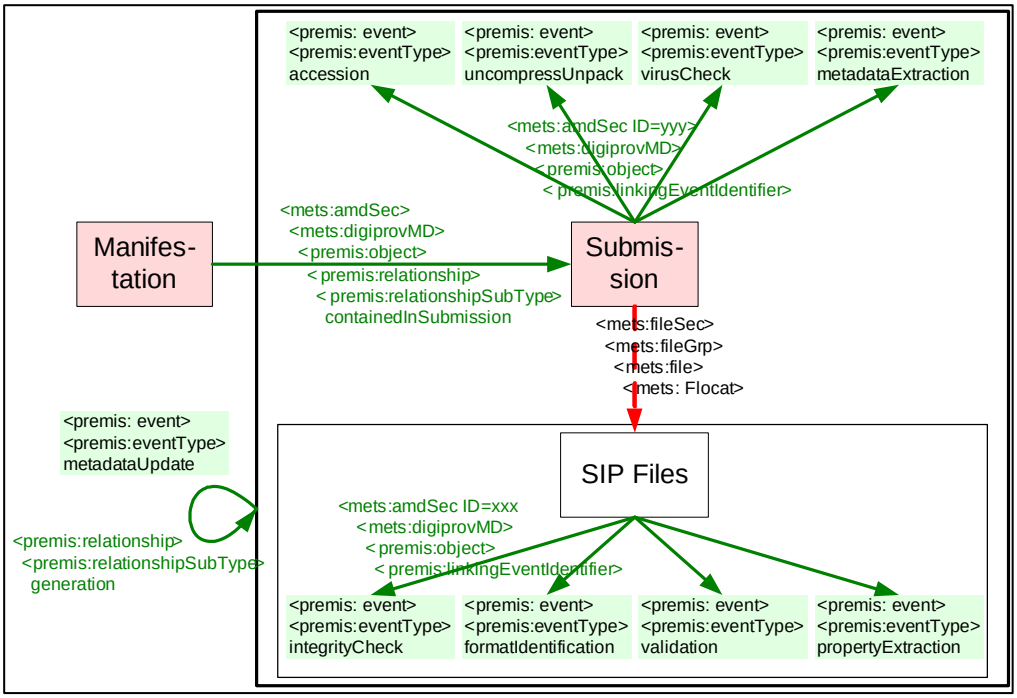

Figure 5: Data Model - Submission

Provenance Metadata for Submissions

Similarly to structural entities and manifestations, a submission object can have a "generation" <premis: relationship> with a "metadataUpdate" <premis:event>.
All other events recorded for a submission object are free of side-effects. Events such as "accession", "uncompressUnpack", "metadataExtraction" and "virusCheck" should theoretically be recorded on a file level, but are actually recorded at the submission level in order to avoid redundancy, since they are identical for all files of a submission.

Provenance Metadata for SIPs

The <mets:amdSec> associated with each SIP file has the same side-effect free events as were described for content files in manifestations. It does not contain any events with side-effects or relationships to other files.

METS, PREMIS and MODS Trade-offs

Several metadata elements can be represented in either or several of the metadata schemas. When choosing between them it is helpful to consider that the purpose of the metadata schemas are very different. METS describes a document, while PREMIS stores preservation data for the document or for certain parts (files) of it, and MODS captures descriptive information. Some of the metadata that is captured can be used for several different purposes.
Basic technical metadata, for example, such as checksums and file sizes, are important for preservation purposes but are also part of a complete and detailed description of a digital document. Appropriate elements are available in both schemas (for example,
<premis:object><premis:objectCharacteristic>
<premis:size> and
<mets:fileSec><mets:fileGrp><mets:file SIZE=…>
as well as
<premis:object><premis:objectCharacteristic>
<premis:fixity><premis:messageDigest> and
<mets:fileSec><mets:fileGrp>
<mets:file CHECKSUM=…>)
It is envisaged that this information is used in use cases that access either the METS or the PREMIS metadata portions separately. We therefore store this identical information redundantly in METS and PREMIS. Additionally, it was desirable to be able to store several checksums in a repeatable element, such as in PREMIS, rather than in a non-repeatable attribute, such as offered by METS.
For file format information our considerations were as follows. While METS only stores the MIME-type of a file, PREMIS permits referencing an external format registry. For eJournals the PRONOM database is used and referenced. The MIME-type is usually sufficient to disseminate and render a file (e.g., the MIME type needs to be incorporated in the http-header when transferring files). But for preservation purposes further information about the file format, such as the version or used compression algorithm, might be very important. In theory the MIME type could be extracted from the PRONOM registry, but every dissemination would require a request to the PRONOM database. Storing the data redundantly is, therefore, convenient, especially as there was no concern about data becoming inconsistent in our write-once archival store. While the <mods:physicalDescription> element offers the possibility of specifying technical properties, we decided to keep all technical metadata together in METS or PREMIS where they would be used together. Using the relevant MODS subelements offered no advantage over the more fit-for-purpose elements in PREMIS and METS. We therefore chose not to use MODS on a manifestation or file level at all.
Even though the relationship between a manifestation object and its article object can be regarded as a hierar-

chical one and could be recorded via a <mods:relatedItem> link, we did not want to introduce a MODS section for manifestations just for holding it. Instead the "manifestationOf" <premis:relationship> element is used within the administrative metadata section.
Rights information in our AIPs is not intended to be actionable, in the sense that it does not directly support any repository function, such as access or preservation. Rather it is of an archival, descriptive nature. We, therefore, capture it in MODS rather than PREMIS in order to keep it together with other descriptive information. MODS rights information that is of an administrative nature and might change but is still considered archival, such as embargo information, is stored in the <mets:amdSec><mets:rightsMD> section, whereas descriptive rights information, such as the persistent copyright statement, is kept with other descriptive metadata in the <mets:dmdSec>.
An event that affects several objects is recorded in each affected object's <mets:amdSec>. To create a complete set of metadata, the related agent - the software that executed the event - is stored within the same <mets:amdSec>. Unlike proposed in the current version of the METS-PREMIS guidelines ([PREMIS June 2008]), the <premis:agent> is stored redundantly within each PREMIS section of the same METS file. As each PREMIS section contains a complete set of metadata for a file, extracting, storing or indexing it for preservation purposes becomes very easy.

PREMIS 1.0 versus 2.0

Our current implementation uses version 1.1 of the PREMIS data dictionary ([Premis 2005]) and the corresponding XML schema. After version 2.0 was released in March 2008 ([PREMIS March 2008], [Lavoie 2008]), the impact of changes on the current AIP format were investigated.
Neither the fundamental data model of PREMIS, nor the event and relationship information have changed. The most important change is the possibility of using extensions from within PREMIS that permit embedding of metadata from other metadata schemas. Some elements used in the AIPs could be refined within PREMIS using an additional metadata schema. The event outcome, as well as the creating application, the object characteristics, and the significant properties could be described in more detail.
For us, the <premis:objectCharacteristicsExtension> might beneficially be used to capture further, object or format-specific, technical metadata for a file. Currently this data is stored in a <mets:techMD> technical metadata section using the JHOVE schema. If it is only used for preservation purposes, it might be useful to move it to the <premis:objectCharacteristicsExtension> instead.
Bigger changes have been made in the XML schema. It does not only support the additional elements, but also defines abstract <premis:object> types, and creates special instances for *representation*, *file* and *bitstream*. These instances allow the mapping of the data dictionary's applicability and obligation constraints to the XML schema and ties them to the object type. This might improve and simplify the validation process

Conclusion

No single existing metadata schema accommodates the representation of descriptive, preservation and structural metadata. This paper shows how we use a combination of METS, PREMIS and MODS to represent eJournal Archival Information Packages in a write-once archival system.

References

METS 2006. *Metadata Encoding and Transmission Standard (METS) Official Web Site. Version 1.6.* http://www.loc.gov/standards/mets/

PREMIS Working Group, May 2005. *Data Dictionary for Preservation Metadata: Final Report of the Premis Working Group, version 1.0.* http://www.oclc.org/research/projects/pmwg/premis-final.pdf

MODS 2006. *Metadata Object Description Schema. Version 3.2.* http://www.loc.gov/standards/mods/

Bray, T. et al. eds. 2006, *Extensible Markup Language (XML) 1.0 (Fourth Edition).* http://www.w3.org/TR/2006/REC-xml-20060816/

CCSDS, January 2002. *Reference Model for an Open Archival Information System (OAIS). CCSDS 650.0-B-1, Blue Book* (the full ISO standard). http://public.ccsds.org/publications/archive/650x0b1.pdf

IFLA Study Group on the Functional Requirements for Bibliographic Records, 1998. Functional requirements for bibliographic records : final report. München: K.G. Saur, 1998. (UBCIM publications; new series, vol. 19). ISBN 3-598-11382-X.

PREMIS Editorial Committee, March 2008. *PREMIS Data Dictionary for Preservation Metadata, version 2.0.* http://www.loc.gov/standards/premis/v2/premis-2-0.pdf

PREMIS in METS Working Group, June 2008 *Guidelines for using PREMIS with METS for exchange*. Revised June 25, 2008. http://www.loc.gov/standards/premis/guidelines-premismets.pdf

Guenther, R., 2008. *Battle of the Buzzwords; Flexibility vs. Interoperability When Implementing PREMIS in METS.* D-Lib Magazine, July/August 2008, Vol. 14 No. 7/8, doi:10.1045/dlib.magazine, ISSN: 1082-9873. http://www.dlib.org/dlib/july08/guenther/07guenther.html

Lavoie, B. 2008. *PREMIS with a fresh coat of paint: Highlights from the Revision of the PREMIS Data Dictionary for Preservation Metadata.* D-Lib Magazine, May/June 2008,Vo. 14 No. 5/6, ISSN 1082-9873. http://www.dlib.org/dlib/may08/lavoie/05lavoie .html

This article was also published in D-Lib Magazine, September/October 2008, Vol. 14, No. 9/10.