

## Linking Between and Within METS and PREMIS Documents

Thomas Habing  
University of Illinois at Urbana-Champaign  
thabing@uiuc.edu

This paper assumes the reader is familiar with both the PREMIS and METS standards including the XML schema for each.

### METS

The below is based on METS version 1.8.

#### Linking Between Sections within METS Documents

Linking between the various sections of a METS document is done almost exclusively by means of XML attributes of the ID and IDREF or IDREFS data types. This is especially true in regards to linking between the various sections in the METS document, such as from a file to the metadata about that file, from a structural division to the file which manifests that division, or from descriptive metadata to administrative metadata about it. There are pros and cons to using ID and IDREFS for this. The primary pro is that most XML parsers can automatically validate the linkages, ensuring that there are no duplicate ID values within a single document and ensuring that each IDREF or IDREFS value refers to a corresponding ID value in the same document. This is possible even in the absence of an XML Schema or DTD. There is also some support for this style of identifier in the general Web architecture, for example using fragment identifiers appended to URIs to identify specific sub-sections of a document, such as using `http://some.edu/mets.xml#ADM_12345` to identify in the `mets.xml` file the specific `<amdSec>` with `ID='ADM_12345'`. This means that if a METS document is published to a specific URL, it is easy for external applications or documents to refer to specific sections of the METS document so long as they are assigned an ID attribute.

There are some cons to this approach. One being that the XML ID data type is restricted to a specific syntax and set of characters. Generally it must start with an underscore or letter followed by letters, digits, or some other special characters, such as periods or dashes. For example, this means that most URIs are not valid XML IDs. Another con is that ID and IDREFS values are generally only meaningful within the document in which they occur. It is possible for an implementer to use unique XML IDs across their entire corpus of METS documents, but this would require additional infrastructure and would not really be very useful if the METS documents are shared outside of the local environment.

METS also supports links between the structural divisions of a METS document by means of the XLink schema. For example, this can be used to create hyperlinks between different divisions in a `<structMap>` or to establish relationships between divisions in a logical `<structMap>` to divisions in a

physical <structMap>. These links can be between divisions in the same METS file or to divisions between different METS files.

METS also has multiple attributes and structures that support various sorts of implicit relationships and links. For example, the GROUPID attribute on the <file> element presumably creates a relationship between all the other <file> elements that share that same GROUPID attribute value. The <structMap> elements purpose is to allow the creation of rich relationships and linkages between files or data streams within files.

### External Content in METS

In almost all cases METS allows 'external content' to either be embedded/wrapped within the METS document itself or to be referenced from the METS document. METS also allows the same file to be both embedded and linked with the caveat that they must be the same file. This includes the files which are manifestations of the digital object represented by the METS document as well as the various metadata sections used to describe the digital object. Embedded content can be either XML or encoded binary. For referenced content the xlink:href attribute is used to establish links to external objects.

There are multiple considerations when deciding between linking and embedding, and most METS documents use a combination of the two depending on the content. Often the files which compose the digital object are linked while the various metadata sections are embedded, but this may not be true for all METS profiles.

The advantage of embedding everything is that the entirety of the digital object, including all the content and its metadata, are in a single file which can simplify management of the object and may be desirable for preservation of archival information packages. However, in most situations this can lead to very large, unwieldy, and difficult to process files. The disadvantage of linking to everything is that packages could consist of very many separate files which all must be tracked and managed to avoid broken or obsolescent links. However, in some cases where the METS file creator cannot gain direct access to the file, perhaps because of intellectual property constraints, the only option is to reference the file via an xlink:href.

In addition to links to arbitrary external content, METS also supports a special kind of external link which must point to another METS file. The <mptr> element is used for these links. It is typically used to assert relationships between objects represented by different METS files, for example that one METS file represents a different version of an object in a different METS file, or that an object in one METS file is composed of objects represented in multiple other METS files.

## PREMIS

The below is based on PREMIS version 2.0.

### Linking Between Sections within PREMIS Documents

Like METS, PREMIS also supports internal linking between entities using attributes of the XML ID and IDREF data types. Each of the four PREMIS entities has an xmlID attribute, and all of the linking or related element types have a corresponding Link...XmlID or Rel...XmlID attribute which is an IDREF.

These IDREF must point to one of the xmlID attributes. Using these ID and IDREF attributes for internal linking has the same advantages and disadvantages as already described for METS. However, it also adds the constraint that if ID and IDREF are used, all of the PREMIS entities must be described in a single XML file; otherwise, XML validation will fail.

Unlike in METS, the use of the ID and IDREF attributes is optional. The preferred (I assume preferred because the elements are required) linking mechanism is via a mandatory identifier element associated with each type of PREMIS entity: <objectIdentifier>, <eventIdentifier>, <agentIdentifier>, and <rightsStatementIdentifier>, each consisting of a type and value. Any linking or related element type must then reference one of these identifiers via a mandatory linking or related identifier element, such as these: <linkingEventIdentifier>, <linkingRightsStatementIdentifier>, <linkingAgentIdentifier>, <linkingObjectIdentifier>, <relatedObjectIdentification>, and <relatedEventIdentification>, each also consisting of a type and value.

The advantage of this approach is that the identifiers used can be of any type, such as URIs. Also, using this method, the entities which may have an existence outside of the PREMIS namespace, such as objects or agents, can have more than one identifier, possibly making the location or identification of these entities more robust over time. Using this type of identification schema also allows the various PREMIS entities to be split into separate XML files without worrying that ID and IDREF links will be broken. For example, if the same agent participates in multiple events across multiple PREMIS intellectual entities, the agent only needs to be described once in a PREMIS agent XML file. It can then be referenced from multiple places without the need to copy the entire agent description into each file that references it. The disadvantage of this approach is the overhead required to manage all the identifiers and ensure that links are not broken over time.

Another linking mechanism in the PREMIS XML schema that does not seem to be as well defined as the previous two is the use of XLink Simple Links. Each PREMIS entity identifier element allows xlink:simpleLink attributes to be used. It is (to this author) unclear to what an xlink attached to a PREMIS entity identifier should point. For an object or agent it seems that it could point to a representation of the entity, but PREMIS already has other elements that accomplish this. Using xlink for the event or rights identifiers is equally as unclear, other than possibly pointing to a representation of the entity. However, the linking and related element types also allow for XLink Simple Links which seem like they might be more useful as another alternate method for pointing to linked or related objects. It has an advantage over the IDREF in that it must be a URI, allowing it to point to external files, and it could be a URI with a fragment identifier pointing to a specific ID element in a different file. In any case, you could accomplish the exact same thing by using the already mandatory PREMIS elements.

The identifier and linking and related elements are mandatory. However, the ID and IDREF linking or the XLink-style linking may be combined with this method.

### External Content in PREMIS

PREMIS supports several elements that allow XML content from other arbitrary namespaces to be embedded within a PREMIS XML file. The elements allowing this are these:

<creatingApplicationExtension>, <environmentExtension>, <eventOutcomeDetailExtension>, <keyInformation>, <objectCharacteristicsExtension>, <rightsExtension>, <signatureInformationExtension>, and <significantPropertiesExtension>.

PREMIS also supports elements designed to reference or link to external content. These include the identifiers assigned to the four PREMIS entity types, especially the object and the agent identifiers which may have multiple identifiers, including the xlink:href attribute already mentioned. These values are used to identify these entities, but may double as URLs pointing to the locations of the entity or some representation thereof. The external locations of the objects (files, bitstreams, or representations) may also be indicated by the <originalName> element either as text content or via an xlink:href attribute. The location of the objects as stored in the controlling repository may be indicated by the <contentLocation> element either as text content or via an xlink:href attribute. A representation or surrogate of the intellectual entity whose representation is being preserved may also be identifier or located using <linkingIntellectualEntityIdentifier> element. Finally, external format registries may be identifier or located using the <formatRegistry> element.

### PREMIS and METS Together

Given the above background we now consider how the various PREMIS and METS elements may be used together. A typical scenario uses METS as the wrapper for some sort of archival package, such as an OAIS SIP, DIP, or AIP, with PREMIS used for various aspects of the preservation metadata about the entities in the package.

Probably the two most significant factors to consider are how the PREMIS entities are segregated into one or more sections or files, and whether those sections or files are linked from the METS file or embedded in the METS file. These two factors could be represented as follows:

|   |  |
|---|--|
| One PREMIS File Embedded in the METS File | One Embedded File for each PREMIS Entity |
| One PREMIS File Linked from the METS File | One Linked File for each PREMIS Entity   |

There are variations of the above, for example a mixture of linked and embedded PREMIS files in one METS file, or multiple PREMIS files each containing one object entity and multiple other PREMIS entities all either linked or embedded in the METS file. However, I believe that describing the above four scenarios should illustrate all major advantages and disadvantages of the different approaches.

#### One PREMIS File Embedded in the METS File

The first case to examine will be a single PREMIS file with root element <premis> containing all the preservation metadata about an entire intellectual entity, including object, event, agent, and rights entities. The assumption is that the PREMIS file is internally consistent and could stand alone if required. Therefore, none of the internal links between the various PREMIS entities will be shown in the following diagrams, but they are assumed to be there. This single PREMIS file will be embedded in a METS file which packages all the files and metadata needed to represent that intellectual entity. An


advantage of this approach is that preservation metadata can be easily extracted and used as a stand-alone file, possibly simplifying the management of these data.

Typically the PREMIS file would be embedded in one of the <amdSec> subsections. The assumption being that preservation metadata is administrative more than descriptive. Unfortunately there is no single <amdSec> subsection which seems best for all preservation metadata which can encompass technical, rights, provenance, and source metadata. In any case, the specific <amdSec> subsection is not significant for this discussion. In addition, it is conceivable that the PREMIS file could be treated as just another file being managed within the METS package in which case it could be embedded as <FContent> in a <file> element. Because this case is not in keeping with typical METS practice of separation of data and metadata, it will not be treated separately. However, in actuality the linking challenges associated with this case do not differ significantly from those involved when the PREMIS is embedded in an <amdSec>.

Since the single PREMIS file represents the preservation metadata for the entire intellectual entity represented by the METS file, it would make sense to use the ADMID of the root <div> element of the METS <structMap> to point to the ID of the <amdSec> subsection containing the PREMIS file:

**METSFile.xml:**

```
<mets:amdSec>
  <mets:techMD ID="ALLPreservation001"> ←
    <mets:mdWrap MDTYPE='PREMIS'>
      <mets:xmlData>
        <premis:premis>
<mets:structMap>
  <mets:div ADMID=' ALLPreservation001'> —
```



However, it is less clear how to link specific entities in the PREMIS to the corresponding elements in the METS, such as a METS <file> element to the specific PREMIS <object> element containing the preservation metadata about that file. Given the flexibility of both the PREMIS and METS XML schema there are several options.

### *Using ID and IDREF Attributes*

One solution might be to use the ID and IDREF/IDREFS attributes that are supported by both schemas. For example, each PREMIS <object> element or other entity would have an xmlID attribute and then the METS <file> element could link to these entities via its ADMID attribute:

**METSFile.xml:**

```

<premis:object xsi:type='premis:file' xmlID='OBJ1'>
<premis:rights xmlID='RGTS1'>
<premis:event xmlID='EVT1'>
<mets:file ADMID='OBJ1 EVT1 RGTS1'>

```

Technically this would be allowed by the XML schemas and would have the advantage of leveraging the built-in validation capabilities of XML, and it provides for fine-grained linkages between METS sections and corresponding PREMIS entities. However, this tightly couples the PREMIS and METS documents in such a way that they could not be easily separated. In addition, some might argue that this violates the implicit semantics of METS that requires ADMID attributes to only point to METS <amdSec> subsections.

***Implicit Linking Using PREMIS Identifiers***

A better solution might be to rely on the native identifiers themselves. For example, each PREMIS object must have at least one identifier, and the METS <file> element usually has an <FLocat> with an xlink:href attribute, or it might have an OWNERID attribute, either of which can be used to establish the relationship with the PREMIS object:

**METSFile.xml:**

```

<premis:object xsi:type='premis:file' >
  <premis:objectIdentifier>
    <premis:objectIdentifierType>URL</premis:objectIdentifierType>
    <premis:objectIdentifierValue>http://Host/Path/File.ext</premis:objectIdentifierValue>
  </premis:objectIdentifier>

<mets:file>
  <mets:FLocat LOCTYPE='URL' xlink:href=' http://Host/Path/File.ext' />

or

<mets:file OWNERID=' http://Host/Path/File.ext'>
  <mets:FContent>

```

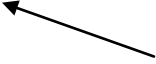
This establishes an implicit relationship via the common identifier values, and has the advantage of utilizing the already mandatory entity identifier elements required by PREMIS, and also of keeping the METS and PREMIS more loosely coupled so they can be separated if needed; however, not using the XML ID and IDREF linking mechanism means that extra effort may be required to avoid broken links.

### *Implicit Linking Using PREMIS <contentLocation> or <originalName> Elements*

As an alternative to the PREMIS identifier elements, the object's <contentLocation> or <originalName> elements could be used to tie an object to its corresponding file in the METS document:

#### **METSFile.xml:**


```
<premis:storage>
  <premis:contentLocation>/Path/File.ext</premis:contentLocation>
</premis:storage>
<mets:file>
  <mets:FLocat LOCTYPE='URL' xlink:href=' /Path/File.ext' />
```



or

#### **METSFile.xml:**

```
<premis:originalName>http://Host/Path/File.ext</premis:originalName>
<mets:file OWNERID=' http://Host/Path/File.ext'>
```



This does not really offer any advantages over using the PREMIS identifiers, but might be preferable if a system wants to avoid mixing identifiers and locators.

You could also combine the above, possibly using the implicit linking provided by using common identifier values in the various PREMIS and METS elements or attributes, while at the same time using the explicit links provided by the XML ID and IDREF attributes.

### **One PREMIS File Linked from the METS File**

This scenario is very similar to the previous; except that the PREMIS file is linked from the METS file instead of being embedded in the METS file.

Because IDREFS must point to an ID in the same document, this immediately precludes using XML ID and IDREF attributes to link from the METS directly into the PREMIS file.

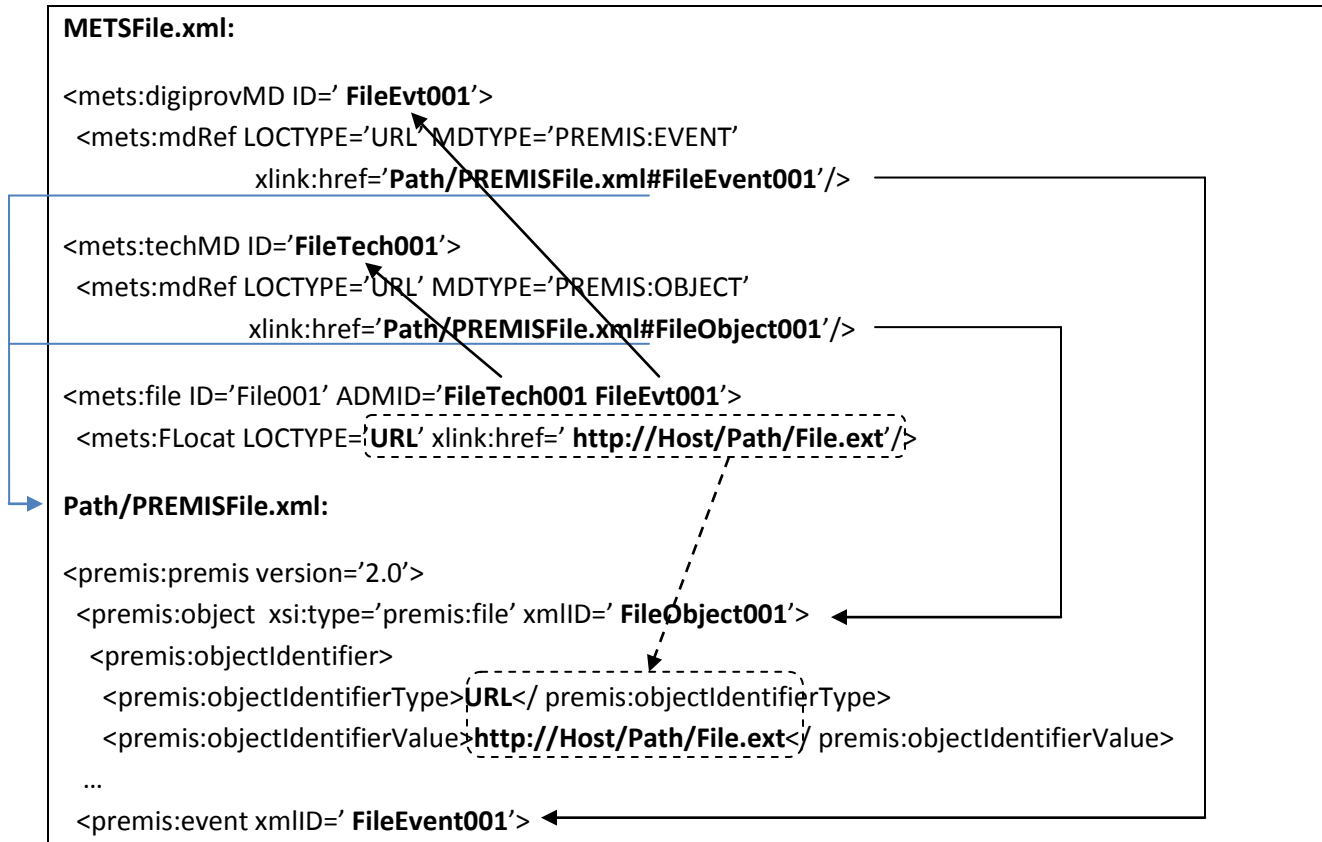
### *Implicit Linking Using PREMIS Identifier, <contentLocation>, or <originalName> Elements*

An obvious solution is to treat the PREMIS exactly as in the previous scenario except that it is linked from instead of embedded in a single <amdSec> subsection. One can then rely on the implicit linking provided by using common identifier values just as described in the previous sections: "Implicit Linking Using PREMIS Identifiers" or "Implicit Linking Using PREMIS <contentLocation> or <originalName> Elements." However, there is another possibility that provides more explicit, fine-grained linking from METS elements to specific PREMIS entities.

### *Linking via URL Fragment Identifiers*

Using this technique the METS is internally structured in typical fashion with specific metadata sections being linked directly from other METS elements via the ID and IDREFS attributes. For example, a <file> element would reference a <techMD> element specific to only that file. However, in this scenario the

<techMD> references a specific object in a PREMIS file via a URL fragment identifier. The PREMIS file is also structured in typical fashion with one addition: each entity that needs to be directly referenced from the METS file must have an xmlID attribute. The xmlID attribute provides the target for the URL fragment identifier. The example below shows a METS file that references a PREMIS object and a PREMIS event, both contained in a single external PREMIS file:



This technique has the advantage of allowing both the METS and PREMIS to be constructed in a typical fashion as standalone files, but it also allows the fairly explicit fine-grained linkages to be made from METS elements to specific, corresponding PREMIS entities. This could also be combined with the more implicit linking provided by common identifiers or locators that are shared between the METS and PREMIS files, as shown above with the dashed lines connecting the URL `http://Host/Path/File.ext` values.

### One Embedded File for each PREMIS Entity & One Linked File for each PREMIS Entity

These two approaches are similar enough that I have combined them into a single section.

This approach allows each PREMIS entity to be a stand-alone file, either embedded or linked from the appropriate METS <amdSec> subsection. This approach has the advantage of allowing explicit, fine-grained links from the METS elements to their corresponding PREMIS entities. It also allows for the different PREMIS entities to be packaged in the most appropriate METS <amdSec> subsection: <object> in <techMD>, <rights> in <rightsMD>, and <event> in <digiprovMD>. For example, a METS file might have some corresponding PREMIS technical, rights, and events metadata which might be represented as shown below:





In the above example, the PREMIS is shown as being referenced, but the links would be very similar if it was instead embedded. Using dashed lines, this example also illustrates the implicit, identifier value-based linking between the different PREMIS entities and between the METS <FLocat> element and the PREMIS object. These links between the different PREMIS entities are assumed to be present in the previous examples, but are shown here because each of the PREMIS entities is in a separate file, potentially making these links more difficult to maintain because the identifier must now be maintained across an entire system and not just within a single PREMIS file.

### Hybrid Combinations

In addition to the above there might also be hybrid solutions. For example, if multiple objects comprise an intellectual entity each object along with all of its corresponding events, rights, and agents might be contained in a single PREMIS file. This would result in multiple PREMIS files, but fewer than if each entity was its own file. This might also simplify management of objects in a repository, but it might result in some duplication, for example, if the same event affects multiple objects requiring that event to be placed in multiple files. In any case, any combination of the above techniques for linking the PREMIS entities to their corresponding METS sections should still work whether PREMIS object files are embedded or linked.

### Conclusions

For better or worse, both PREMIS and METS allow for a multitude of flexible options with regard to making relationships between both internal and external entities. Individual implementers and users will need to weigh the options and choose which linking method or combination of methods best meets their own needs.