Purpose

This specification covers a single discrete object that has many file datastreams. Those file datastreams were originally stored in a hierarchical manner. Preserving and later presenting the associated files using the original hierarchy, filenames, and extensions is required.

The purpose of this specification is to provide a solution for storing hierarchal file structures of a single discrete object.

METS Structural Map

When investigating ways to store hierarchal file structures for preservation and reuse the METS structural map needs consideration. The METS structural map was developed to outline the hierarchal structure of a digital library object. Linking capabilities exist between the elements of the structural map and related content files.

Structural maps encode information in an XML file with the outermost element being <structMap>. Hierarchies are then described as a nested series of division elements, , that carry attributed information that specifies the type of division it is.

Division elements can include multiple METS pointers, <mptr>, and file pointers, <fptr>, that are used to describe associated content. The METS pointers are used to specify separate METS documents that contain relevant file information for the division they are contained under. This is useful for separating very large structural maps into separate files keeping all structural maps relatively small. File pointers specify files or groups of files or specific parts of a file within the METS file section <fileSec>. The following is an example of a simple structural map provided by the Library of Congress.

```
<structMap TYPE="logical">
        <div ID="div1" LABEL="Oral History: Mayor Abraham Beame" TYPE="oral history">
                <div ID="div1.1" LABEL="Interviewer Introduction" ORDER="1">
                        <fptr FILEID="FILE001">
                                <area FILEID="FILE001" BEGIN="INTVWBG" END="INTVWND" BETYPE="IDREF" />
                        </fptr>
                        <fptr FILEID="FILE002">
                                <area FILEID="FILE002" BEGIN="00:00:00" END="00:01:47" BETYPE="TIME" />
                        </fptr>
                        <fptr FILEID="FILE003">
                                <area FILEID="FILE003" BEGIN="00:00:00" END="00:01:47" BETYPE="TIME" />
                        </fptr>
                </div>
                <div ID="div1.2" LABEL="Family History" ORDER="2">
                        <fptr FILEID="FILE001">
                                <area FILEID="FILE001" BEGIN="FHBG" END="FHND" BETYPE="IDREF" />
                        </fptr>
                        <fptr FILEID="FILE002">
                                <area FILEID="FILE002" BEGIN="00:01:48"END="00:06:17" BETYPE="TIME" />
                        </fptr>
                        <fptr FILEID="FILE003">
                                <area FILEID="FILE003" BEGIN="00:01:48" END="00:06:17" BETYPE="TIME" />
                        </fptr>
                </div>
        </div>
</structMap>
```

Source: http://www.loc.gov/standards/mets/METSOverview.v2.html#structmap

METS File Section

The file section in METS, <fileSec>, can contain one or more file group elements, <fileGrp>. The <fileGrp> elements contain a list of all files that make up a digital resource. File group elements are typically used to group together versions of a file. An example might be a file section with two file group, one for archival masters and another for the presentation version. Currently we do not store the METS file section as part of the object in RUCore. We store the object in FOXML, which has a similar capability. Below is an example, provided by the Library of Congress, of a file section with three different version of an oral history.

```
<fileSec>
        <fileGrp ID="VERS1">
                <file ID="FILE001" MIMETYPE="application/xml" SIZE="257537" CREATED="2001-06-10">
                        <FLocat LOCTYPE="URL">http://dlib.nyu.edu/tamwag/beame.xml</FLocat>
                </file>
        </fileGrp>
        <fileGrp ID="VERS2">
                <file ID="FILE002" MIMETYPE="audio/wav" SIZE="64232836" CREATED="2001-05-17" GROUPID="AUDIO1">
                        <FLocat LOCTYPE="URL">http://dlib.nyu.edu/tamwag/beame.wav</FLocat>
                </file>
        </fileGrp>
        <fileGrp ID="VERS3" VERSDATE="2001-05-18">
                <file ID="FILE003" MIMETYPE="audio/mpeg" SIZE="8238866" CREATED="2001-05-18" GROUPID="AUDIO1">
                        <FLocat LOCTYPE="URL">http://dlib.nyu.edu/tamwag/beame.mp3</FLocat>
                </file>
        </fileGrp>
</fileSec>
```

Source: http://www.loc.gov/standards/mets/METSOverview.v2.html#filegrp

Technical Metadata

Currently our technical metadata section stores information pertaining to the archival master files for a resource. We do not store technical information for presentation files. Below is an example of our current technical metadata section.

```
<rulib:RULTechMD xmlns:rulib="http://rucore.libraries.rutgers.edu/schemas/rulib/0.1/metadata.dtd" ID="ARCH1.0">
        <rulib:fileSize UNIT="bytes">11376640</rulib:fileSize>
        <rulib:operatingSystem VERSION="5.1">windows xp</rulib:operatingSystem>
        <rulib:contentModel>ETD</rulib:contentModel>
        <rulib:mimeType TYPE="file">application/pdf</rulib:mimeType>
        <rulib:mimeType TYPE="container">application/x-tar</rulib:mimeType>
        <rulib:fileSize UNIT="bytes">11376640</rulib:fileSize>
        <rulib:checksum METHOD="SHA1">cdd2a1bdf103d0d3c480286cde0d1cb868b8b913</rulib:checksum>
</rulib:RULTechMD>
```

Source: http://rucore.libraries.rutgers.edu/api/get/36563/technical/1

Considerations

In order to consider structural maps as a solution for storing the structure of and delivering the files for objects, developing a file section, using FOXML capability or adapting technical metadata to include all files needs consideration.

If both archival master and presentation file structures are hierarchical will they always follow the same hierarchy, or might those hierarchies differ?

Structural Map Possibilities

If structure maps were used to describe an objects file hierarchy a division, might be considered a directory/folder and a file might be represented using the file pointer, <fptr>, element.

**Division elements ** can have the following attributes that would help describe information about the directory/folder they are representing.

@ID – Unique ID of directory/folder, system generated.

@TYPE – METS places no constraints on vocabulary used, so "folder" might be the logical option.

@LABEL – The actual directory/folder name.

@ORDER – An integer representation of this div's order among its siblings (e.g., its sequence)

@DMDID – ID of related descriptive metadata about the directory/folder.

@ADMID – ID of pertinent administrative metadata about the directory/folder.

File pointers <fptr> can have the following attributes that would be used to describe partial information about the objects files.

@ID – Unique identifier for the element in the METS document, system generated.

@FILEID – Provides the ID used in the file section of the METS document. Associates that file pointer to a file element (@ID) found in the file section.

@CONTENTIDS – A content ID, unique external ID, represented by the file pointer. Fedora datastream IDs could be stored in this attribute.

Consideration

The limiting factor of the structural map is that a `<div>` elements current level in the hierarchy is not stored in a dedicated element.

Option 1: Append the structural map schema by adding a @LEVEL attribute that would be an integer value that stores the current level of the `<div>` element. Any editing or manipulation of the hierarchy would cause re-writing of this attribute value and possible others depending on how extensive the changes were, i.e. changing a `<div>` elements level would cause its attribute value and any children attributes levels to be edited to maintain consistency.

The @LEVEL attribute could then be removed if the structural map was to be shared/exported with external partners. This would make the stored structural map non-METS compliant as a @LEVEL attribute does not currently exist in the schema.

Option 2: Leverage dot notation in the @ID attribute to store hierarchy information along with the natural `<div>` element order in the XML document. The dot notation could take the form `<level>.<unique integer>`. Examples are:

Level 1, Division 1 would be @ID="1.1"

Level 1, Division 2 would be @ID="1.2"

Level 2, Division 1 would be @ID="2.1"

Editing of the structural maps hierarchy would require a re-writing of all affected @IDs to maintain consistency. Using this notation would not violate the METS schema and we could directly export the structural map to external partners without having to modify it.

Option 2 appears the most viable for long term support of the structural maps. The order and level are stored in two ways, using the dot notation of the identifier attribute and the nested structure of the XML document itself. No additional processing will need to be performed in the event the structural maps are shared with external partners.

Source: http://www.loc.gov/standards/mets/mets.xsd

File Section Possibilities

If the METS file section were used in tandem with a structural map file group elements with multiple file elements inside of it could be used. If both archival and presentation hierarchies were to be preserved two file groups could be considered, one for archival and one for presentation.

File group elements <fileGrp> can use the following attributes to describe each of those logical groupings.

@ID – Unique ID of the grouping, system generated.

@VERSDATE – Version date of the grouping if we wish to support multiple versions of the grouping.

@ADMID - ID of pertinent administrative metadata about the grouping.

@USE – Used to indicate the intended use of the group, the terms “archival” and presentation” are admissible.

File elements <file> appear under the grouping and are related to corresponding entries in the structural map via a system generated ID. The following attributes might also be used to describe files in this element.

@ID - Unique identifier for the element in the METS document, system generated. Associates the element to a file pointer(@FILEID) that is stored in the structural map.

@MIMETYPE – The mime-type of file.

@SIZE – Size of the file in bytes.

@CREATED – Creation date of file.

@CHECKSUM – Checksum value for the file.

@CHECKSUMTYPE – Type of checksum value.

@ADMID - ID of pertinent administrative metadata about the file.

@DMDID – ID of related descriptive metadata about the file.

@USE – Indicated the use of the file, e.g master, thumbnail etc.

A **file location <FLocat>** element is present for each file element. That element could use the following attributes to further describe the file.

@ID – Unique identifier for the element in the METS document, system generated.

@LOCTYPE – The locator type used in the xlink:href attribute. Valid values for LOCTYPE are: *ARK, URN, URL, PURL, HANDLE, DOI, OTHER.*

@OTHERLOCTYPE – Specifies the locater type when @LOCTYPE is *OTHER*.

@xlink:href – URI or other information indicating where the content file is located.

@xlink:title – Human readable title of the file. This could be the place to store the original file name and extension.

FOXML Investigation

Since objects are currently store in FOXML in RUcore an investigation needs to be performed to see if existing FOXML capability could provide the same functionality as the METS file section. Examining the FOXML schema reveals the following attributes are legal and acceptable for the **<datastreamVersion>** element.

@ID – *required* – datastream ID

@MIMETYPE – *required* – mime-type of datastream

@LABEL – *optional* – User friendly label for datastream

@CREATED – *optional* – creation date

@SIZE - *optional* – size of datastream in bytes

@FORMAT_URI – *optional* - user-assigned URI used to identify the media type of the bytestream (more specific than MIME type). *Should we consider using this for curation purposes?*

@ALT_IDS – *optional* - user-assigned set of alternative identifiers for the datastream, with the identifiers separated by spaces. *Should we consider this for original filename so long as we accept the space delimitation is not imposed?*

Source: http://fedora-commons.org/definitions/1/0/foxml1-1.xsd

Example - Archival Structural Map

The following is an example of how an objects archival hierarchical characteristics might be stored in common RUcore object. In this example a complex file hierarchy was created that is composed for sample sub-directories and files.

```
C:\example
|   text.txt (1KB | 4/4/2012 9:32 AM)
|
├───Folder A
|   |   METSPrimerRevised.pdf (1,570KB | 3/21/2012 11:51 AM)
|   |   text.txt (1KB | 3/21/2012 12:33 PM)
|   |
|   └───Folder A.1
|           leeroy jenkins.mp3 (90KB | 3/8/2012 1:31 PM)
```

```
|
├──Folder B
|       nuclear_full.png (437KB | 12/9/2009 3:39 PM)
|
└──Folder C
        random_stuff.tar (15,315KB | 4/4/2012 9:34 AM)
```

We begin by modeling the file information into corresponding archival <foxml:datastreamVersion> elements for the FOXML object. Please note the inclusion of a DIGEST attribute in the <foxml:contentDigest> element. This is included under the assumption that checksum values are being moved to the greater FOXML document from the technical metadata datastream.

```
<foxml:datastream ID="ARCH1" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="ARCH1.0" LABEL="Explanation of package" CREATED="2012-04-04T09:32:24.000Z"
MIMETYPE="text/plain" SIZE="18" FORMAT_URI="" ALT_IDS="text.txt">
       <foxml:contentDigest TYPE="SHA-1" DIGEST="1ce17160c5bad3ab977b57f7f15d2f6c1b31a924"/>
       <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+ARCH1+ARCH1.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>

<foxml:datastream ID="ARCH2" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="ARCH2.0" LABEL="METS Primer document" CREATED="2012-03-21T11:51:14.000Z"
MIMETYPE="application/pdf" SIZE="1607557" FORMAT_URI="" ALT_IDS="METSPrimerRevised.pdf">
       <foxml:contentDigest TYPE="SHA-1" DIGEST="8413c524ce4d9fc9f0d05c0a971b809416ea045a"/>
       <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+ARCH2+ARCH2.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>

<foxml:datastream ID="ARCH3" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="ARCH3.0" LABEL="METS Primer addendum" CREATED="2012-03-21T12:33:45.000Z"
MIMETYPE="text/plain" SIZE="22" FORMAT_URI="" ALT_IDS="text.txt">
          <foxml:contentDigest TYPE="SHA-1" DIGEST="e8425f3de134312560981611a8c8aee143b96506"/>
          <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+ARCH3+ARCH3.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>

<foxml:datastream ID="ARCH4" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="ARCH4.0" LABEL="Excerpt of the primer in audio form" CREATED="2012-03-
08T13:31:07.000Z" MIMETYPE="audio/mpeg" SIZE="91438" FORMAT_URI="" ALT_IDS="leeroy jenkins.mp3">
          <foxml:contentDigest TYPE="SHA-1" DIGEST="2fa9e330112aa38a64bd9d44948a729f32d82e5f"/>
          <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+ARCH4+ARCH4.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>

<foxml:datastream ID="ARCH5" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="ARCH5.0" LABEL="Nuclear Explosions since 1945.  It's cool, trust me!"
CREATED="2009-12-09T15:39:12.000Z" MIMETYPE="image/x-png" SIZE="447293" FORMAT_URI=""
ALT_IDS="nuclear_full.png">
       <foxml:contentDigest TYPE="SHA-1" DIGEST="ddae4c36ca21391ac9c3a6ab2b440fd9439d07c3"/>
       <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+ARCH5+ARCH5.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>

<foxml:datastream ID="ARCH6" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
```

```
    <foxml:datastreamVersion ID="ARCH6.0" LABEL="Helper files" CREATED="20021-04-04T09:34:58.000Z"
MIMETYPE="application/tar" SIZE="15682197" FORMAT_URI="" ALT_IDS="random_stuff.tar">
            <foxml:contentDigest TYPE="SHA-1" DIGEST="7e60ffd4c8c54c173a905c77ff4053bfe94a8f8f"/>
            <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+ARCH6+ARCH6.0"/>
        </foxml:datastreamVersion>
</foxml:datastream>
```

A functional structural map of the hierarchy for archival purposes needs to be modeled as well. This might be stored as a separated managed datastream with using the SMAP-ARCH datastream ID.

```
<structMap TYPE="logical">
        <div ID="div1" LABEL="Example" ORDER="1" TYPE="folder">
                <fptr FILEID="FILE001" CONTENTIDS="ARCH1"/>
                <div ID="div1.1" LABEL="Folder A" ORDER="1" TYPE="folder">
                        <fptr FILEID="FILE002" CONTENTIDS="ARCH2"/>
                        <fptr FILEID="FILE003" CONTENTIDS="ARCH3"/>
                        <div ID="div1.1.1" LABEL="Folder A.1" ORDER="1" TYPE="folder">
                                <fptr FILEID="FILE004" CONTENTIDS="ARCH4"/>
                        </div>
                </div>
                <div ID="div1.2" LABEL="Folder B" ORDER="2" TYPE="folder">
                        <fptr FILEID="FILE005" CONTENTIDS="ARCH5"/>
                </div>
                <div ID="div1.3" LABEL="Folder C" ORDER="3" TYPE="folder">
                        <fptr FILEID="FILE006" CONTENTIDS="ARCH6"/>
                </div>
        </div>
</structMap>
```

Example - Presentation Structural Map

In this example we reuse the sample file hierarchy from the archival structural map example, however some of the files that were archived are not intended for display.

```
C:\example

|   text.txt (1KB | 4/4/2012 9:32 AM)

|

├──Folder A

|   |   METSPrimerRevised.pdf (1,570KB | 3/21/2012 11:51 AM) [archive only]

|   |   text.txt (1KB | 3/21/2012 12:33 PM)

|   |

|   └──Folder A.1

|           leeroy jenkins.mp3 (90KB | 3/8/2012 1:31 PM)

|

├──Folder B

|       nuclear_full.png (437KB | 12/9/2009 3:39 PM) [archive only]

|
```

```
└────Folder C

        random_stuff.tar (15,315KB | 4/4/2012 9:34 AM)
```

Once again we begin by modeling the file information into corresponding archival <foxml:datastreamVersion> elements for the FOXML object. In this example though let us consider that we have persistent URL's that can point to individual presentation datastreams. The only logical place to store that persistent URL is in the ALT_IDS attribute along with original file name.

```
<foxml:datastream ID="TXT-1" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="TXT-1.0" LABEL="Explanation of package" CREATED="2012-04-04T09:32:24.000Z"
MIMETYPE="text/plain" SIZE="18" FORMAT_URI=""
ALT_IDS="fname:text.txt|purl:http://hdl.rutgers.edu/1782.1/dataset.m000086532.txt-1">
       <foxml:contentDigest TYPE="SHA-1" DIGEST="1ce17160c5bad3ab977b57f7f15d2f6c1b31a924"/>
       <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+TXT-1+TXT-1.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>

<foxml:datastream ID="TXT-2" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="TXT-2.0" LABEL="METS Primer addendum" CREATED="2012-03-21T12:33:45.000Z"
MIMETYPE="text/plain" SIZE="22" FORMAT_URI="" ALT_IDS="fname:text.txt|
purl:http://hdl.rutgers.edu/1782.1/dataset.m000086532.txt-2">
       <foxml:contentDigest TYPE="SHA-1" DIGEST="e8425f3de134312560981611a8c8aee143b96506"/>
       <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+TXT-2+TXT-2.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>


<foxml:datastream ID="MP3-1" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="MP3-1.0" LABEL="Excerpt of the primer in audio form" CREATED="2012-03-
08T13:31:07.000Z" MIMETYPE="audio/mpeg" SIZE="91438" FORMAT_URI="" ALT_IDS="fname:leeroy
jenkins.mp3|purl:http://hdl.rutgers.edu/1782.1/Dataset.m000086532.mp3-1">
       <foxml:contentDigest TYPE="SHA-1" DIGEST="2fa9e330112aa38a64bd9d44948a729f32d82e5f"/>
       <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+MP3-1+MP3-1.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>

<foxml:datastream ID="TAR-1" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="TAR-1.0" LABEL="Helper files" CREATED="20021-04-04T09:34:58.000Z"
MIMETYPE="application/tar" SIZE="15682197" FORMAT_URI="" ALT_IDS="fname:random_stuff.tar|
purl:http://hdl.rutgers.edu/1782.1/Dataset.m000086532.tar-1">
          <foxml:contentDigest TYPE="SHA-1" DIGEST="7e60ffd4c8c54c173a905c77ff4053bfe94a8f8f"/>
          <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:12345+TAR-1+TAR-1.0"/>
       </foxml:datastreamVersion>
</foxml:datastream>
```

Multiple ALT_IDS are separated by a pipe |. To support ALT_ID's that might have valid spaces in the value an important addition has been made. That addition is the use of an ALT_ID namespace which is separated from the value it supports through the use of a colon.

For the TXT-1 datastream the ALT_IDS attributes can be examined as follows.

ALT_IDS="**fname:text.txt|purl:http://hdl.rutgers.edu/1782.1/dataset.m000086532.txt-1**"

fname denotes original file name, which has the value text.txt

purl denotes a persistent URL, which has the value http://hdl.rutgers.edu/1782.1/dataset.m000086532.txt-1

Additional alternative ID's could be supported in a similar way.

A functional structural map of the hierarchy for presentation purposes needs to be modeled as well. This might be stored as a separated managed datastream with using the SMAP-PRES datastream ID.

```
<structMap TYPE="logical">
	<div ID="div1" LABEL="Example" ORDER="1" TYPE="folder">
		<fptr FILEID="FILE001" CONTENTIDS="TXT-1"/>
		<div ID="div1.1" LABEL="Folder A" ORDER="1" TYPE="folder">
			<fptr FILEID="FILE002" CONTENTIDS="TXT-2"/>
			<div ID="div1.1.1" LABEL="Folder A.1" ORDER="1" TYPE="folder">
				<fptr FILEID="FILE003" CONTENTIDS="MP3-1"/>
			</div>
		</div>
		<div ID="div1.2" LABEL="Folder C" ORDER="2" TYPE="folder">
			<fptr FILEID="FILE004" CONTENTIDS="TAR-1"/>
		</div>
	</div>
</structMap>
```

Workflow Management System Use Case Scenarios

The process of adding an item to RUcore begins with the Workflow Management System, WMS. The following are use case scenarios for adding different types of files in both simple and complex ways to an item in the WMS.

Scenario One – Single TIFF file

Story

An item is being created that has one TIFF image file associated with it. The cataloger will upload the file from either their computer or using the server upload functionality.

Expectations

A default is set wither application-wide or at the collection level that determines if the original filename needs to be preserved for the archival master.

Derivative/presentation files are generated. In this case a THUMBJPEG-1, JPEG-1, and PDF-1.

Since this is a simple object with one file no archival structural map or presentation structural map is generated.

Question

When publically accessing the derivative/presentation files should the files be delivered to the end user using the original filename?

Answer

If the original filename is recorded in the ALT_ID attribute it should be used when the file is delivered through a public interface. If a filename value is not stored a system generated filename based on the object ID and datastream ID will be generated. This will support earlier resources whose original filenames have not been recorded.

Scenario Two – Multiple TIFF files

Story

An item is being created that has multiple TIFF image files associated with it. The TIFF files are uploaded in a structured manner with digital master and master directories. The cataloger will upload the file from either their computer or using the server upload capability in WMS.

Expectations

Since there are multiple files associated with this item the user and the files are presented to the WMS in a structured manner the WMS will automatically generate an archival structure map.

Derivative/presentation files are generated. In this case a THUMBJPEG-1, JPEG-1, and PDF-1. No structural map for the presentation files is generated.

Question

Does it present an issue that an archival structural map was generated, while a presentation structural map was not?

Answer

No it does not present an issue. This is an expected behavior in some cases.

Scenario Three – Directory of Files

Story

A cataloger has identified a directory of files they wish to attach to an item. The cataloger will use the server upload capability in WMS.

Expectations

Original files names are preserved in the ALT_ID attribute and will be used when delivering the files through a public interface.

User has the ability to select a directory and all files and sub-directories that are part of that parent directory.

Since there are multiple files associated with this item the user and the files are presented to the WMS in a structured manner the WMS will automatically generate an archival structure map.

Some of the files that were uploaded need to have derivative/presentation version generated. Some files do not, and they can be presented in their original format. Some files should not be publically accessible at all. A presentation structural map is generated.

Scenario Four – Document Submitted Through the Faculty Deposit Module

Story

A user wishes to deposit a document through the faculty deposit module.

Expectations

User is not asked if the original filename should be preserved when accessing it at a later date. This decision should be made for them, as a system default.

Derivative/presentation files are generated. In this case a THUMBJPEG-1 and PDF-1.

Since this is a simple object with one file no archival structural map or presentation structural map is generated.

Scenario Five – Yearbooks

Story

A user wishes to digitize and store a yearbook.

Expectations

User is not asked if the original filename should be preserved when accessing it at a later date. This decision should be made for them, as a system default.

Derivative/presentation files are generated.

Since this is a simple object with multiple files stored in a flat manner no archival structural map or presentation structural map is generated.

Scenario Six – Finding Aids/EADS

Story

A user wishes to store a findings aids EAD XML.

Expectations

User is not asked if the original filename should be preserved when accessing it at a later date. This decision should be made for them, as a system default.

Derivative/presentation files are generated.

Since this is a simple object with a single EAD XML datastream no archival structural map or presentation structural map is generated. Digital objects that are associated and used in the finding aid are stored ass separate objects and they are referenced using RDF ontologies via Fedora RELS-EXT technology.

Scenario Seven – Websites

Story

A user wishes to store a website.

Expectations

Due to the complex nature of websites further investigation will need to occur. A data model will need to be defined that will support website. If it were decided that an entire website would be stored as a single Fedora object, then generating archival and presentation structural maps would be required. Those structural maps would then preserve and present the websites original file structure.

Scenario Eight – Multi-volume Sets and Serials

Story

A user wishes to digitize and store a multi-volume set or serial.

Expectations

Due to the complex nature of the both multi-volume sets and serials further an exhaustive data model investigation will need to occur before establishing a workflow decision.

Implementation Decisions

Workflow Management System

- For every object that is newly created in WMS an archival structural map (SMAP-ARCH) will be created. This is true for simple objects with no file hierarchy, in which case the archival structural map will appear flat.
- If a directory of files is presented to WMS the entire directory and contents will be used as archival masters.
- If a WMS user wants to upload a directory of files from his/her workstation they must upload the directory as a packaged file, i.e. ZIP, TAR etc. WMS will then prompt the user and ask if the file should be expanded and its contents archived separately.
- The WMS will offer a feature to edit an ingested structural map of either type, archival or presentation.

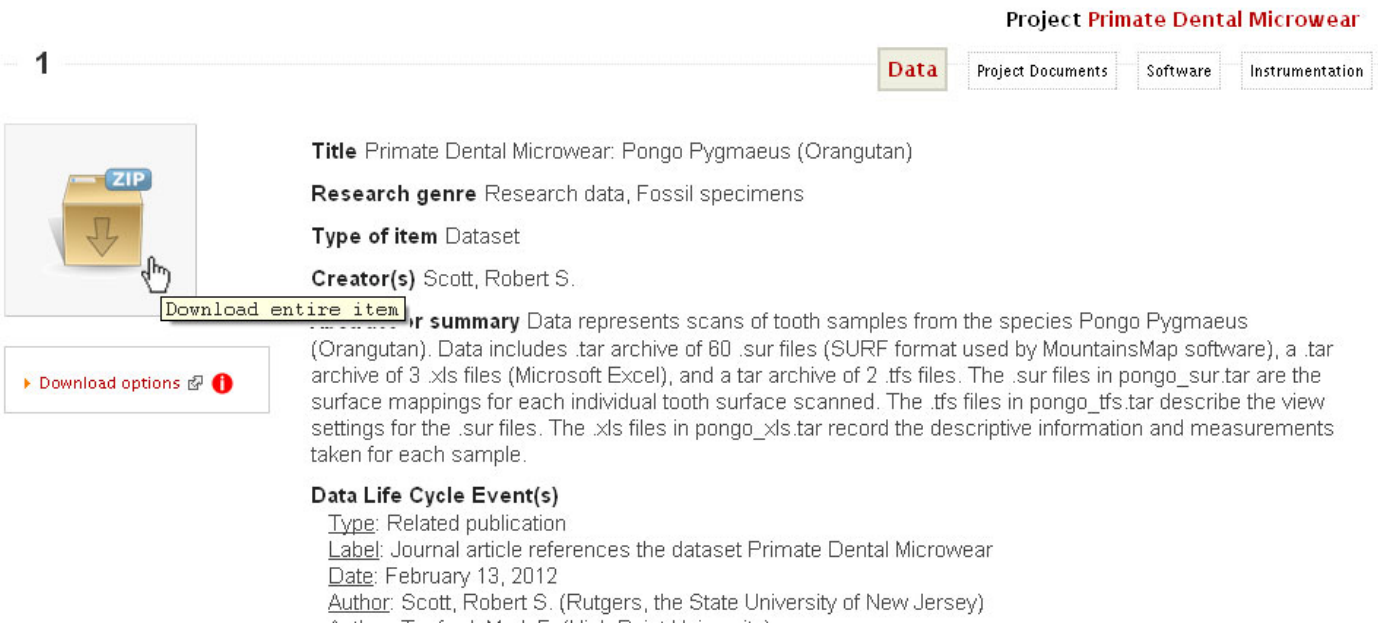
dlr/EDIT

- Editing of the raw XML of the archival and presentation structural maps will be disabled in dlr/EDIT.

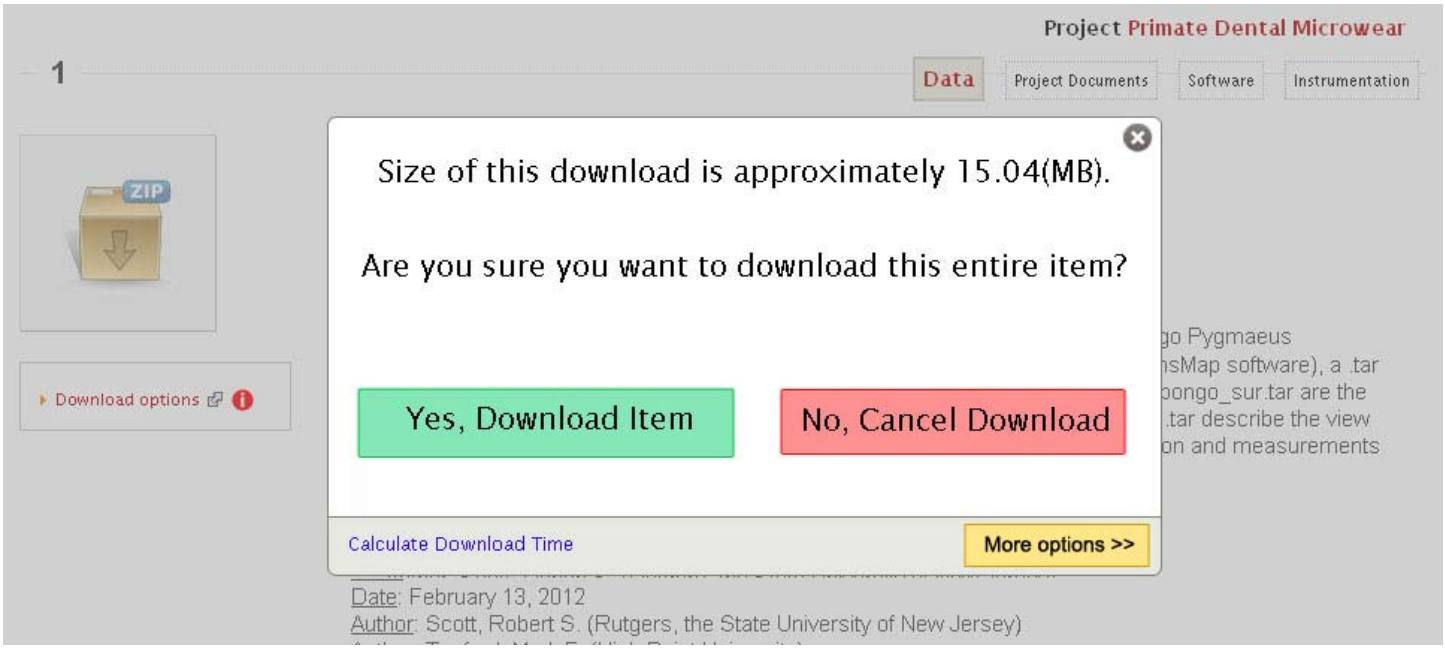
Example - User Experience for Downloading a File Hierarchy

When a user discovers a resource in the repository that contains a presentation structural map as part of its datastream profile an appropriate user experience must be rendered. This experience must include an intuitive way to download part of or the complete object. The following are a series of screen renderings of the user experience.

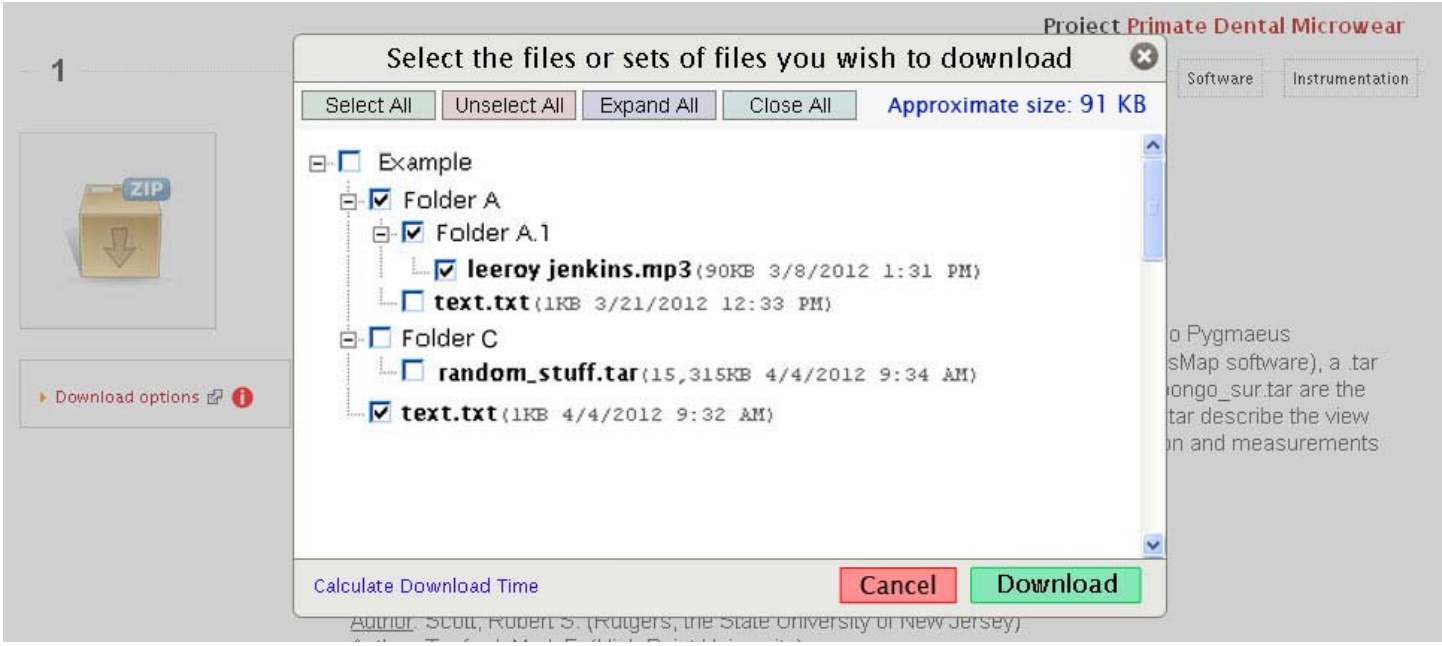
In the following screen rendering a user discovers a resource that has a presentation structural map as part of the objects datastream makeup. Based on that characteristic the following full record result is rendered. Note the ability to “Download entire item” by selecting the ZIP icon or by selecting the “Download options” link.



Selecting the "**Download options**" link could present a module window for confirmation. In the window the entire size of the download would be present along with the option to calculate the time it would take to download the entire item.



Selecting the "**More options**" button could present a module window offering the user selection choices. In the screen rendering below the presentation structural map example is represented. Some files have been chosen. Notice as files are selected the "**Approximate size**" is recalculated. Also the option to "**Calculate Download Time**" is available. If selected that would report the time in minutes and second and also adjust based on the size of the selected files. In this example the original file names are used in the file selection display. Optionally we can use the designated user friendly label, if supplied, in the file selection display.



Once the user selects to "**Download**" a background process will need to package up all of the files selected. The package that is created will maintain original files names and directory structure.

It is the suggestion of this specification that the BagIt specification developed at the Library of Congress be considered as the file package standard, http://www.digitalpreservation.gov/documents/bagitspec.pdf. From the Library of Congress's site the following is a brief description of BagIt.

> *A specification for the packaging of digital content for transfer. Content is packaged (the bag) along with a small amount of machine-readable text (the tag) to help automate the content's receipt, storage and retrieval. There is no software to install. A bag consists of a base directory containing the tag and a subdirectory that holds the content files. The tag is a simple text-file manifest, like a packing slip, that consists of two elements:*
>
> *1. An inventory of the content files in the bag*
> *2. A checksum for each file.*

Source: http://www.digitalpreservation.gov/tools/#b