DSpace AIP Format

1 Makeup and Definition of AIPs
- 1.1 AIPs are Archival Information Packages.
- 1.2 General AIP Structure / Examples
  - 1.2.1 Customizing What Is Stored in Your AIPs

2 AIP Details: METS Structure

3 Metadata in METS
- 3.1 DIM (DSpace Intermediate Metadata) Schema
  - 3.1.1 DIM Descriptive Elements for Item objects
  - 3.1.2 DIM Descriptive Elements for Collection objects
  - 3.1.3 DIM Descriptive Elements for Community objects
  - 3.1.4 DIM Descriptive Elements for Site objects
- 3.2 MODS Schema
- 3.3 AIP Technical Metadata Schema (AIP-TECHMD)
  - 3.3.1 AIP Technical Metadata for Item
  - 3.3.2 AIP Technical Metadata for Bitstream
  - 3.3.3 AIP Technical Metadata for Collection
  - 3.3.4 AIP Technical Metadata for Community
  - 3.3.5 AIP Technical Metadata for Site
- 3.4 PREMIS Schema
  - 3.4.1 PREMIS Metadata for Bitstream
- 3.5 DSPACE-ROLES Schema
  - 3.5.1 Example of DSPACE-ROLES Schema for a SITE AIP
  - 3.5.2 Example of DSPACE-ROLES Schema for a Community or Collection
- 3.6 METSRights Schema
  - 3.6.1 Example of METSRights Schema for an Item
  - 3.6.2 Example of METSRights Schema for a Collection
  - 3.6.3 Example of METSRights Schema for a Community

Makeup and Definition of AIPs

AIPs only store the Latest Version of Items

If you are using the new XMLUI-only Item Level Versioning functionality (disabled by default), you must be aware that this "Item Level Versioning" feature is **not yet compatible** with AIP Backup & Restore. **Using them together may result in accidental data loss.** Currently the AIPs that DSpace generates only store the *latest version* of an Item. Therefore, past versions of Items will always be lost when you perform a restore / replace using AIP tools.

AIPs are Archival Information Packages.

- AIP is a package describing **one archival object** in DSpace.
  - The **archival object** may be a single **Item**, **Collection**, **Community**, or **Site** (Site AIPs contain site-wide information). Bitstreams are included in an Item's AIP.
  - Each AIP is logically self-contained, can be restored without rest of the archive. (So you could restore a single Item, Collection or Community)
  - Collection or Community AIPs do **not** include all child objects (e.g. Items in those Collections or Communities), as each AIP only describes **one** object. However, these container AIPs do contain references (links) to all child objects. These references can be used by DSpace to automatically restore all referenced AIPs when restoring a Collection or Community.
  - AIPs are only generated for objects which are currently in the "in archive" state in DSpace. This means that in-progress, uncompleted submissions are not described in AIPs and cannot be restored after a disaster. Permanently removed objects will also no longer be exported as AIPs after their removal. However, withdrawn objects will continue to be exported as AIPs, since they are still considered under the "in archive" status.
  - AIPs with identical contents will always have identical checksums. This provides a basic means of validating whether the contents within an AIP have changed. For example, if a Collection's AIP has the same checksum at two different points in time, it means that Collection has not changed during that time period.
  - AIP profile favors completeness and accuracy rather than presenting the semantics of an object in a standard format. It conforms to the quirks of DSpace's internal object model rather than attempting to produce a universally understandable representation of the object. When possible, an AIP tries to use common standards to express objects.
  - An AIP *can* serve as a DIP (Dissemination Information Package) or SIP (Submission Information Package), especially when transferring custody of objects to another DSpace implementation.
  - In contrast to SIP or DIP, the AIP should include all available DSpace structural and administrative metadata, and basic provenance information. AIPs also describe some basic system level information (e.g. Groups and People).

General AIP Structure / Examples

Generally speaking, an AIP is an Zip file containing a METS manifest and all related content bitstreams, license files and any other associated files.

Some examples include:

- Site AIP (Sample: SITE-example.zip)
  - METS contains basic metadata about DSpace Site and persistent IDs referencing all Top Level Communities
  - METS also contains a list of all Groups and EPeople information defined in the DSpace system. (NOTE: By default, user passwords are not stored in AIPs, unless you specify the 'passwords' flag. See Additional Packager Options.)
- Community AIP (Sample: COMMUNITY@123456789-1.zip)

    - METS contains all metadata for Community and persistent IDs referencing all members (SubCommunities or Collections). Package may also include a Logo file, if one exists.
    - METS contains any Group information for Commmunity-specific groups (e.g. `COMMUNITY_<ID>_ADMIN` group).
    - METS contains all Community permissions/policies (translated into METSRights schema)
- Collection AIP (Sample: COLLECTION@123456789-2.zip)
    - METS contains all metadata for Collection and persistent IDs referencing all members (Items). Package may also include a Logo file, if one exists.
    - METS contains any Group information for Collection-specific groups (e.g. `COLLECTION_<ID>_ADMIN`, `COLLECTION_<ID>_SUBMIT`, etc.).
    - METS contains all Collection permissions/policies (translated into METSRights schema)
    - If the Collection has an Item Template, the METS will also contain all the metadata for that Item Template.
- Item AIP (Sample: ITEM@123456789-8.zip)
    - METS contains all metadata for Item and references to all Bundles and Bitstreams. Package also includes all Bitstream files.
    - METS contains all Item/Bundle/Bitstream permissions/policies (translated into METSRights schema)

Notes:

- Bitstreams and Bundles are second-class archival objects; they are recorded in the context of an Item.
- BitstreamFormats are not even second-class; they are described implicitly within Item technical metadata, and reconstructed from that during restoration
- EPeople are only defined in Site AIP, but may be referenced from Community or Collection AIPs
- Groups may be defined in Site AIP, Community AIP or Collection AIP. Where they are defined depends on whether the Group relates specifically to a single Community or Collection, or is just a general site-wide group.

What is NOT in AIPs

- DSpace Site configurations ([dspace]/config/ directory) or customizations (themes, stylesheets, etc) are not described in AIPs
- DSpace Database model (or customizations therein) is not described in AIPs
- Any objects which are not currently in the "In Archive" state are not described in AIPs. This means that in-progress, unfinished submissions are never included in AIPs.

Customizing What Is Stored in Your AIPs

If you choose, you can customize exactly what information is stored in your AIPs. However, you should be aware that you can only restore information which is stored within your AIPs. If you choose to remove information from your AIPs, you will be unable to restore it later on (unless you are also backing up your entire DSpace database and assetstore folder).

AIP Recommendations

It is recommended to minimally use the default settings when generating AIPs. DSpace can only restore information that is included within an AIP. Therefore, if you choose to no longer include some information in an AIP, DSpace will no longer be able to restore that information from an AIP backup

There are two ways to go about customizing your AIP format:

1. You can customize your `dspace.cfg` settings pertaining to AIP generation. These configurations will allow you to specify exactly which DSpace Crosswalks will be called when generating the AIP METS manifest.
2. You can export your AIPs using one of the special options/flags.

AIP Details: METS Structure

This METS Structure is based on the structure decided for the original AipPrototype, developed as part of the MIT & UCSD PLEDGE project.

- `mets` element
    - `@PROFILE` fixed value="http://www.dspace.org/schema/aip/1.0/mets.xsd" (this is how we identify an AIP manifest)
    - `@OBJID` URN-format persistent identifier (i.e. Handle) if available, or else a unique identifier. (e.g. "hdl:123456789/1")
    - `@LABEL` title if available
    - `@TYPE` DSpace object type, one of "DSpace ITEM", "DSpace COLLECTION", "DSpace COMMUNITY" or "DSpace SITE".
    - `@ID` is a globally unique identifier, built using the Handle and the Object type (e.g. `dspace-COLLECTION-hdl:123456789/3`).
- `mets/metsHdr` element
    - `@LASTMODDATE` last-modified date for a DSpace Item, or nothing for other objects.
    - `agent` element:
        - `@ROLE` = "CUSTODIAN",
        - `@TYPE` = "OTHER",
        - `@OTHERTYPE` = "DSpace Archive",
        - `name` = *Site handle*. (Note: The Site Handle is of the format `[handle_prefix]/0`, e.g. "123456789/0")
    - `agent` element:
        - `@ROLE` = "CREATOR",
        - `@TYPE` = "OTHER",
        - `@OTHERTYPE` = "DSpace Software",
        - `name` = "DSpace [version]" (Where "[version]" is the specific version of DSpace software which created this AIP, e.g. "1.7.0")
- `mets/dmdSec` element(s)
    - By default, two `dmdSec` elements are included for all AIPs:
        1. object's descriptive metadata crosswalked to MODS (specified by `mets/dmdSec/mdWrap@MDTYPE="MODS"`). See #MODS Schema section below for more information.
        2. object's descriptive metadata in DSpace native DIM intermediate format, to serve as a complete and precise record for restoration or ingestion into another DSpace. Specified by `mets/dmdSec/mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="DIM"`. See #DIM (DSpace Intermediate Metadata) Schema section below for more information.

    - For Collection AIPs, additional dmdSec elements may exist which describe the Item Template for that Collection. Since an Item template is not an actual Item (i.e. it only includes metadata), it is stored within the Collection AIP. The Item Template's dmdSec elements will be referenced by a `div @TYPE="DSpace ITEM Template"` in the METS `structMap`.
    - When the `mdWrap` `@TYPE` value is `OTHER`, the element *MUST* include a value for the `@OTHERTYPE` attribute which names the crosswalk that produced (or interprets) that metadata, e.g. `DIM`.
- `mets/amdSec` element(s)
    - One or more amdSec elements are include for all AIPs. The first amdSec element contains administrative metadata (technical, source, rights, and provenance) for the entire archival object. Additional amdSec elements may exist to describe parts of the archival object (e.g. Bitstreams or Bundles in an Item).
        - `techMD` elements. By default, two types of `techMD` elements may be included:
            - `PREMIS` metadata about an object may be included here (*currently only specified for Bitstreams (files)*). Specified by `mdWrap@MDTYPE="PREMIS"`. See #PREMIS Schema section below for more information.
            - `DSPACE-ROLES` metadata may appear here to describe the Groups or EPeople related to this object (_currently only specified for Site, Community and Collection). Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="DSPACE-ROLES"`. See #DSPACE-ROLES Schema section below for more information.
        - `rightsMD` elements. By default, there are four possible types of `rightsMD` elements which may be included:
            - `METSRights` metadata may appear here to describe the permissions on this object. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="METSRIGHTS"`. See #METSRights Schema section below for more information.
            - `DSpaceDepositLicense` if the object is an Item and it has a deposit license, it is contained here. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="DSpaceDepositLicense"`.
            - `CreativeCommonsRDF` If the object is an Item with a Creative Commons license expressed in RDF, it is included here. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="CreativeCommonsRDF"`.
            - `CreativeCommonsText` If the object is an Item with a Creative Commons license in plain text, it is included here. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="CreativeCommonsText"`.
        - `sourceMD` element. By default, there is only one type of `sourceMD` element which may appear:
            - `AIP-TECHMD` metadata may appear here. This stores basic technical/source metadata about in object in a DSpace native format. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="AIP-TECHMD"`. See #AIP Technical Metadata Schema (AIP-TECHMD) section below for more information.
        - `digiprovMD` element.
            - *Not used at this time.*
- `mets/fileSec` element
    - For ITEM objects:
        - Each distinct Bundle in an Item goes into a `fileGrp`. The `fileGrp` has a `@USE` attribute which corresponds to the Bundle name.
        - Bitstreams in bundles become `file` elements under `fileGrp`.
        - `mets/fileSec/fileGrp/file`elements
            - Set `@SIZE` to length of the bitstream. There is a redundant value in the <techMD> but it is more accessible here.
            - Set `@MIMETYPE`, `@CHECKSUM`, `@CHECKSUMTYPE` to corresponding bitstream values. There is redundant info in the <techMD>. (For DSpace, the `@CHECKSUMTYPE="MD5"` at all times)
            - SET `@SEQ` to bitstream's SequenceID if it has one.
            - SET `@ADMID` to the list of <amdSec> element(s) which describe this bitstream.
    - For COLLECTION and COMMUNITY objects:
        - *Only* if the object has a *logo bitstream*, there is a `fileSec` with one `fileGrp` child of `@USE="LOGO"`.
        - The `fileGrp` contains one `file` element, representing the logo Bitstream. It has the same `@MIMETYPE`, `@CHECKSUM`, `@CHECKSUMTYPE` attributes as the Item content bitstreams, but does NOT include metadata section references (e.g. `@ADMID`) or a `@SEQ` attribute.
        - See the main `structMap` for the `fptr` reference to this logo file.
- `mets/structMap` - Primary structure map, `@LABEL="DSpace Object", @TYPE="LOGICAL"`
    - For ITEM objects:
        1. Top-Level `div` with `@TYPE="DSpace Object Contents"`.
            - For every Bitstream in Item it contains a `div` with `@TYPE="DSpace BITSTREAM"`. Each Bitstream `div` has a single `fptr` element which references the bitstream location.
        - If Item has primary bitstream, put it in `structMap/div/fptr` (i.e. directly under the `div` with `@TYPE="DSpace Object Contents"`)
    - For COLLECTION objects:
        1. Top-Level `div` with `@TYPE="DSpace Object Contents"`.
            - For every Item in the Collection, it contains a `div` with `@TYPE="DSpace ITEM"`. Each Item `div` has up to two child `mptr`elements:
                a. One linking to the Handle of that Item. Its `@LOCTYPE="HANDLE"`, and `@xlink:href` value is the raw Handle.
                b. (Optional) one linking to the location of the local AIP for that Item (if known). Its `@LOCTYPE="URL"`, and `@xlink:href` value is a relative link to the AIP file on the local filesystem.
            - If Collection has a Logo bitstream, there is an `fptr` reference to it in the very first `div`.
            - If the Collection includes an Item Template, there will be a `div` with `@TYPE="DSpace ITEM Template"` within the very first `div`. This `div @TYPE="DSpace ITEM Template"` must have a `@DMDID` specified, which links to the dmdSec element(s) that contain the metadata for the Item Template.
    - For COMMUNITY objects:
        1. Top-Level `div` with `@TYPE="DSpace Object Contents"`.
            - For every Sub-Community in the Community it contains a `div` with `@TYPE="DSpace COMMUNITY"`. Each Community `div` has up to two `mptr`elements:
                a. One linking to the Handle of that Community. Its `@LOCTYPE="HANDLE"`, and `@xlink:href` value is the raw Handle.
                b. (Optional) one linking to the location of the local AIP file for that Community (if known). Its `@LOCTYPE="URL"`, and `@xlink:href` value is a relative link to the AIP file on the local filesystem.
            - For every Collection in the Community there is a `div` with `@TYPE="DSpace COLLECTION"`. Each Collection `div` has up to two `mptr`elements:
                a. One linking to the Handle of that Collection. Its `@LOCTYPE="HANDLE"`, and `@xlink:href` value is the raw Handle.

            b. (Optional) one linking to the location of the local AIP file for that Collection (if known). Its `@LOCTYPE="URL"`, and `@xlink:href` value is a relative link to the AIP file on the local filesystem.
        - If Community has a Logo bitstream, there is an `fptr` reference to it in the very first `div`.
    - For SITE objects:
        1. Top-Level `div` with `@TYPE="DSpace Object Contents"`.
            - For every Top-level Community in Site, it contains a `div` with `@TYPE="DSpace COMMUNITY"`. Each Item `div` has up to two child `mptr`elements:
                a. One linking to the Handle of that Community. Its `@LOCTYPE="HANDLE"`, and `@xlink:href` value is the raw Handle.
                b. (Optional) one linking to the location of the local AIP for that Community (if known). Its `@LOCTYPE="URL"`, and `@xlink:href` value is a relative link to the AIP file on the local filesystem.
- `mets/structMap` - Structure Map to indicate object's Parent, `@LABEL="Parent", @TYPE="LOGICAL"`
    - Contains one `div` element which has the unique attribute value `TYPE="AIP Parent Link"` to identify it as the older of the *parent pointer*.
        - It contains a `mptr` element whose `xlink:href` attribute value is the raw Handle of the parent object, e.g. `1721.1/4321`.

Metadata in METS

The following tables describe how various metadata schemas are populated (via DSpace Crosswalks) in the METS file for an AIP.

DIM (DSpace Intermediate Metadata) Schema

DIM Schema is essentially a way of representing DSpace internal metadata structure in XML. DSpace's internal metadata is very similar to a Qualified Dublin Core in its structure, and is primarily meant for descriptive metadata. However, DSpace's metadata allows for custom elements, qualifiers or schemas to be created (so it is extendable to any number of schemas, elements, qualifiers). These custom fields/schemas may or may not be able to be translated into normal Qualified Dublin Core. So, the DIM Schema must be able to express metadata schemas, elements or qualifiers which may or may not exist within Qualified Dublin Core.

In the METS structure, DIM metadata always appears within a `dmdSec` inside an `<mdWrap MDTYPE="OTHER" OTHERMDTYPE="DIM">` element. For example:

```
<dmdSec ID="dmdSec_2190">
   <mdWrap MDTYPE="OTHER" OTHERMDTYPE="DIM">
    ...
   </mdWrap>
</dmdSec>
```

By default, DIM metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.dmd = MODS, DIM
```

DIM Descriptive Elements for Item objects

As all DSpace Items already have user-assigned DIM (essentially Qualified Dublin Core) metadata fields, those fields are just exported into the DIM Schema within the METS file.

DIM Descriptive Elements for Collection objects

For Collections, the following fields are translated to the DIM schema:

DIM Metadata Field	Database field or value
dc.description	'introductory_text' field
dc.description.abstract	'short_description' field
dc.description.tableofcontents	'side_bar_text' field
dc.identifier.uri	Collection's handle
dc.provenance	'provenance_description' field
dc.rights	'copyright_text' field
dc.rights.license	'license' field
dc.title	'name' field

DIM Descriptive Elements for Community objects

For Communities, the following fields are translated to the DIM schema:

DIM Metadata Field	Database field or value
dc.description	'introductory_text' field
dc.description.abstract	'short_description' field
dc.description.tableofcontents	'side_bar_text' field
dc.identifier.uri	Handle of Community
dc.rights	'copyright_text' field
dc.title	'name' field

DIM Descriptive Elements for Site objects

For the Site Object, the following fields are translated to the DIM schema:

Metadata Field	Value
dc.identifier.uri	Handle of Site (format: `[handle_prefix]/0`)
dc.title	Name of Site (from dspace.cfg 'dspace.name' config)

MODS Schema

By default, all DSpace descriptive metadata (DIM) is also translated into the MODS Schema by utilizing DSpace's `MODSDisseminationCrosswalk`. DSpace's DIM to MODS crosswalk is defined within your `[dspace]/config/crosswalks/mods.properties` configuration file. This file allows you to customize the MODS that is included within your AIPs.

For more information on the MODS Schema, see http://www.loc.gov/standards/mods/mods-schemas.html

In the METS structure, MODS metadata always appears within a `dmdSec` inside an `<mdWrap MDTYPE="MODS">` element. For example:

```
<dmdSec ID="dmdSec_2189">
   <mdWrap MDTYPE="MODS">
    ...
   </mdWrap>
</dmdSec>
```

By default, MODS metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.dmd = MODS, DIM
```

The MODS metadata is included within your AIP to support interoperability. It provides a way for other systems to interact with or ingest the AIP without needing to understand the DIM Schema. You may choose to disable MODS if you wish, however this may decrease the likelihood that you'd be able to easily ingest your AIPs into a non-DSpace system (unless that non-DSpace system is able to understand the DIM schema). When restoring/ingesting AIPs, DSpace will always first attempt to restore DIM descriptive metadata. Only if no DIM metadata is found, will the MODS metadata be used during a restore.

AIP Technical Metadata Schema (AIP-TECHMD)

The AIP Technical Metadata Schema is a way to translate technical metadata about a DSpace object into the DIM Schema. It is kept separate from DIM as it is considered technical metadata rather than descriptive metadata.

In the METS structure, AIP-TECHMD metadata always appears within a `sourceMD` inside an `<mdWrap MDTYPE="OTHER" OTHERMDTYPE="AIP-TECHMD">` element. For example:

```
<amdSec ID="amd_2191">
    ...
    <sourceMD ID="sourceMD_2198">
       <mdWrap MDTYPE="OTHER" OTHERMDTYPE="AIP-TECHMD">
       ...
       </mdWrap>
    </sourceMD>
    ...
</amdSec>
```

By default, AIP-TECHMD metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.sourceMD = AIP-TECHMD
```

AIP Technical Metadata for Item

Metadata Field	Value
dc.contributor	Submitter's email address
dc.identifier.uri	Handle of Item
dc.relation.isPartOf	Owning Collection's Handle (*as a URN*)
dc.relation.isReferencedBy	All other Collection's this item is linked to (*Handle URN of each non-owner*)
dc.rights.accessRights	*"WITHDRAWN"* if item is withdrawn

AIP Technical Metadata for Bitstream

Metadata Field	Value
dc.title	Bitstream's name/title
dc.title.alternative	Bitstream's source
dc.description	Bitstream's description
dc.format	Bitstream Format Description
dc.format.medium	Short Name of Format
dc.format.mimetype	MIMEType of Format
dc.format.supportlevel	System Support Level for Format (necessary to recreate Format during restore, if the format isn't know to DSpace by default)
dc.format.internal	Whether Format is internal (necessary to recreate Format during restore, if the format isn't know to DSpace by default)

- Outstanding Question: Why are we recording the file format support status? That's a DSpace property, rather than an Item property. Do DSpace instances rely on objects to tell them their support status?
    - Possible answer (from Larry Stone): Format support and other properties of the BitstreamFormat are recorded here in case the Item is restored in an empty DSpace that doesn't have that format yet, and the relevant bits of the format entry have to be reconstructed from the AIP. --lcs

AIP Technical Metadata for Collection

Metadata Field	Value
dc.identifier.uri	Handle of Collection
dc.relation.isPartOf	Owning Community's Handle (*as a URN*)
dc.relation.isReferencedBy	All other Communities this Collection is linked to (*Handle URN of each non-owner*)

AIP Technical Metadata for Community

Metadata Field	Value
dc.identifier.uri	Handle of Community
dc.relation.isPartOf	Handle of Parent Community (*as a URN*)

AIP Technical Metadata for Site

Metadata Field	Value
dc.identifier.uri	Site Handle (format: `[handle_prefix]/0`)

PREMIS Schema

At this point in time, the PREMIS Schema is only used to represent technical metadata about DSpace Bitstreams (i.e. Files). The PREMIS metadata is generated by DSpace's `PREMISCrosswalk`. Only the PREMIS Object Entity Schema is used.

In the METS structure, PREMIS metadata always appears within a `techMD` inside an `<mdWrap MDTYPE="PREMIS">` element. PREMIS metadata is **always** wrapped withn a `<premis:premis>` element. For example:

```
<amdSec ID="amd_2209">
    ...
    <techMD ID="techMD_2210">
       <mdWrap MDTYPE="PREMIS">
          <premis:premis>
            ...
          </premis:premis>
       </mdWrap>
    </techMD>
    ...
</amdSec>
```

Each Bitstream (file) has its own `amdSec` within a METS manifest. So, there will be a separate PREMIS `techMD` for each Bitstream within a single Item.

By default, PREMIS metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.techMD = PREMIS, DSPACE-ROLES
```

PREMIS Metadata for Bitstream

The following Bitstream information is translated into PREMIS for each DSpace Bitstream (file):

Metadata Field	Value
<premis:objectIdentifier>	Contains Bitstream direct URL
<premis:objectCategory>	Always set to "File"
<premis:fixity>	Contains MD5 Checksum of Bitstream
<premis:format>	Contains File Format information of Bistream
<premis:originalName>	Contains original name of file

DSPACE-ROLES Schema

All DSpace Groups and EPeople objects are translated into a custom `DSPACE-ROLES` XML Schema. This XML Schema is a very simple representation of the underlying DSpace database model for Groups and EPeople. The `DSPACE-ROLES` Schemas is generated by DSpace's `RoleCrosswalk`.

Only the following DSpace Objects utilize the DSPACE-ROLES Schema in their AIPs:

- Site AIP – all Groups and EPeople are represented in DSPACE-ROLES Schema
- Community AIP – only Community-based groups (e.g. `COMMUNITY_1_ADMIN`) are represented in DSPACE-ROLES Schema
- Collection AIP – only Collection-based groups (e.g. `COLLECTION_2_ADMIN`, `COLLECTION_2_SUBMIT`, etc.) are represented in DSPACE-ROLES Schema

In the METS structure, DSPACE-ROLES metadata always appears within a `techMD` inside an `<mdWrap MDTYPE="OTHER" OTHERMDTYPE="DSPACE-ROLES">` element. For example:

```
<amdSec ID="amd_2068">
    ...
    <techMD ID="techMD_2070">
       <mdWrap MDTYPE="OTHER" OTHERMDTYPE="DSPACE-ROLES">
            ...
       </mdWrap>
    </techMD>
    ...
</amdSec>
```

By default, DSPACE-ROLES metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.techMD = PREMIS, DSPACE-ROLES
```

Example of DSPACE-ROLES Schema for a SITE AIP

Below is a general example of the structure of a DSPACE-ROLES XML file, as it would appear in a SITE AIP.

```
<DSpaceRoles>
  <Groups>
    <Group ID="1" Name="Administrator">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="0" Name="Anonymous" />
    <Group ID="70" Name="COLLECTION_hdl:123456789/57_ADMIN">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="75" Name="COLLECTION_hdl:123456789/57_DEFAULT_READ">
      <MemberGroups>
        <MemberGroup ID="0" Name="Anonymous" />
      </MemberGroups>
    </Group>
    <Group ID="71" Name="COLLECTION_hdl:123456789/57_SUBMIT">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="72" Name="COLLECTION_hdl:123456789/57_WORKFLOW_STEP_1">
      <MemberGroups>
        <MemberGroup ID="1" Name="Administrator" />
      </MemberGroups>
    </Group>
    <Group ID="73" Name="COLLECTION_hdl:123456789/57_WORKFLOW_STEP_2">
      <MemberGroups>
        <MemberGroup ID="1" Name="Administrator" />
      </MemberGroups>
    </Group>
    <Group ID="8" Name="COLLECTION_hdl:123456789/6703_DEFAULT_READ" />
    <Group ID="9" Name="COLLECTION_hdl:123456789/2_ADMIN">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
  </Groups>
  <People>
    <Person ID="1">
      <Email>bsmith@myu.edu</Email>
      <Netid>bsmith</Netid>
      <FirstName>Bob</FirstName>
      <LastName>Smith</LastName>
      <Language>en</Language>
      <CanLogin />
    </Person>
    <Person ID="2">
      <Email>jjones@myu.edu</Email>
      <FirstName>Jane</FirstName>
      <LastName>Jones</LastName>
      <Language>en</Language>
      <CanLogin />
      <SelfRegistered />
    </Person>
  </People>
</DSpaceRoles>
```

Why are there Group Names with Handles?

You may have noticed several odd looking group names in the above example, where a Handle is embedded in the name (e.g. "COLLECTION_hdl:123456789/57_SUBMIT"). This is a translation of a Group name which included a Community or Collection *Internal ID* (e.g. "COLLECTION_45_SUBMIT"). Since you are exporting these Groups outside of DSpace, the *Internal ID* may no longer be valid or be understandable. Therefore, before export, these Group names are all translated to include an externally understandable identifier, in the form of a Handle. If you use this AIP to restore your groups later, they will be translated back to the normal DSpace format (i.e. the handle will be translated back to the new *Internal ID*).

Orphaned Groups are Renamed on Export

If a Group name includes a Community or Collection *Internal ID* (e.g. "COLLECTION_45_SUBMIT"), and that Community or Collection no longer exists, then the Group is considered "Orphaned".

- In 1.8.2 and above, the Group is renamed using the following format: "ORPHANED_[object-type]_GROUP_[obj-id]_[group-type]" (e.g. "ORPHANED_COLLECTION_GROUP_10_ADMIN").
- Prior to 1.8.2, the Group was renamed with a random key: "GROUP_[random-hex-key]_[object-type]_[group-type]" (e.g. "GROUP_123eb3a_COLLECTION_ADMIN"). *This old format was discontinued as giving the groups a randomly generated name caused the SITE AIP to have a different checksum every time it was regenerated (see DS-1120).*

The reasoning is that we were unable to translate an *Internal ID* into an *External ID* (i.e. Handle). If we are unable to do that translation, re-importing or restoring a group with an *old* internal ID could cause conflicts or instability in your DSpace system. In order to avoid such conflicts, these groups are renamed using a random, unique key.

Example of DSPACE-ROLES Schema for a Community or Collection

Below is a general example of the structure of a DSPACE-ROLES XML file, as it would appear in a Community or Collection AIP.

This specific example is for a Collection, which has associated Administrator, Submitter, and Workflow approver groups. In this very simple example, each group only has one Person as a member of it. Please notice that the Person's information (Name, NetID, etc) is NOT contained in this content (however they are available in the DSPACE-ROLES example for a SITE, as shown above)

```
<DSpaceRoles>
  <Groups>
    <Group ID="9" Name="COLLECTION_hdl:123456789/2_ADMIN" Type="ADMIN">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="13" Name="COLLECTION_hdl:123456789/2_SUBMIT" Type="SUBMIT">
      <Members>
        <Member ID="2" Name="jjones@myu.edu" />
      </Members>
    </Group>
    <Group ID="10" Name="COLLECTION_hdl:123456789/2_WORKFLOW_STEP_1" Type="WORKFLOW_STEP_1">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="11" Name="COLLECTION_hdl:123456789/2_WORKFLOW_STEP_2" Type="WORKFLOW_STEP_2">
      <Members>
        <Member ID="2" Name="jjones@myu.edu" />
      </Members>
    </Group>
    <Group ID="12" Name="COLLECTION_hdl:123456789/2_WORKFLOW_STEP_3" Type="WORKFLOW_STEP_3">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
  </Groups>
</DSpaceRoles>
```

METSRights Schema

All DSpace Policies (permissions on objects) are translated into the METSRights schema. This is different than the above DSPACE-ROLES schema, which only represents Groups and People objects. Instead, the METSRights schema is used to translate the permission statements (e.g. a group named "Library Admins" has Administrative permissions on a Community named "University Library"). But the METSRights schema doesn't represent who is a member of a particular group (that is defined in the DSPACE-ROLES schema, as described above).

METSRights should always be used with DSPACE-ROLES

The METSRights Schema must be used in conjunction with the DSPACE-ROLES Schema for Groups, People and Permissions to all be restored properly. As mentioned above, the METSRights metadata can only be used to restore permissions (i.e. DSpace policies). The DSPACE-ROLES metadata must also exist if you wish to restore the actual Group or EPeople objects to which those permissions apply.

All DSpace Object's AIPs (except for the SITE AIP) utilize the METSRights Schema in order to define what permissions people and groups have on that object. Although there are several sections to the METSRights Schema, DSpace AIPs *only use* the `<RightsDeclarationMD>` section, as this is what is used to describe rights on an object.

In the METS structure, METSRights metadata always appears within a `rightsMD` inside an `<mdWrap MDTYPE="OTHER" OTHERMDTYPE="METSRIGHTS">` element. For example:

```
  <amdSec ID="amd_2068">
      ...
      <rightsMD ID="rightsMD_2074">
         <mdWrap MDTYPE="OTHER" OTHERMDTYPE="METSRIGHTS">
              ...
         </mdWrap>
      </rightsMD>
      ...
  </amdSec>
```

By default, METSRights metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.rightsMD = DSpaceDepositLicense:DSPACE_DEPLICENSE, \
    CreativeCommonsRDF:DSPACE_CCRDF, CreativeCommonsText:DSPACE_CCTEXT, METSRIGHTS
```

Example of METSRights Schema for an Item

An Item AIP will almost always contain several METSRights metadata sections within its METS Manifest. A separate METSRights metadata section is used to describe the permissions on:

- the Item itself
- each Bundle (group of files) in the Item
- each Bitstream (file) within an Item's bundle

Below is an example of a METSRights sections for a publicly visible Bitstream, Bundle or Item. Notice it specifies that the "GENERAL PUBLIC" has the permission to DISCOVER or DISPLAY this object.

```
<rights:RightsDeclarationMD xmlns:rights="http://cosimo.stanford.edu/sdr/metsrights/" RIGHTSCATEGORY="LICENSED">
  <rights:Context CONTEXTCLASS="GENERAL PUBLIC">
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
</rights:RightsDeclarationMD>
```

As of DSpace 3, DSpace policies/permissions may also have a "start-date" or "end-date" (to support Embargo functionality). Such a policy on an Item may look like this. Notice it specifies that the "GENERAL PUBLIC" has the permission to DISCOVER or DISPLAY this object *starting on* 2015-01-01, while the Group "Staff" has permission to DISCOVER or DISPLAY this object *until* 2015-01-01.

```
<rights:RightsDeclarationMD xmlns:rights="http://cosimo.stanford.edu/sdr/metsrights/" RIGHTSCATEGORY="LICENSED">
  <rights:Context CONTEXTCLASS="GENERAL PUBLIC" start-date="2015-01-01" in-effect="false">
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP" end-date="2015-01-01" in-effect="true">
    <rights:UserName USERTYPE="GROUP">Staff</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
</rights:RightsDeclarationMD>
```

Example of METSRights Schema for a Collection

A Collection AIP contains one METSRights section, which describes the permissions different Groups or People have within the Collection

Below is an example of a METSRights sections for a publicly visible Collection, which also has an Administrator group, a Submitter group, and a group for each of the three DSpace workflow approval steps. You'll notice that each of the groups is provided with very specific permissions within the Collection. Submitters & Workflow approvers can "ADD CONTENTS" to a collection (but cannot delete the collection). Administrators have full rights.

```
<rights:RightsDeclarationMD xmlns:rights="http://cosimo.stanford.edu/sdr/metsrights/" RIGHTSCATEGORY="LICENSED">
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_SUBMIT</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="true" DELETE="false" OTHER="true"
```

```
OTHERPERMITTYPE="ADD CONTENTS" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_WORKFLOW_STEP_3</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="true" DELETE="false" OTHER="true"
OTHERPERMITTYPE="ADD CONTENTS" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_WORKFLOW_STEP_2</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="true" DELETE="false" OTHER="true"
OTHERPERMITTYPE="ADD CONTENTS" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_WORKFLOW_STEP_1</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="true" DELETE="false" OTHER="true"
OTHERPERMITTYPE="ADD CONTENTS" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_ADMIN</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" COPY="true" DUPLICATE="true" MODIFY="true" DELETE="true"
PRINT="true" OTHER="true" OTHERPERMITTYPE="ADMIN" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="GENERAL PUBLIC">
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
</rights:RightsDeclarationMD>
```

Example of METSRights Schema for a Community

A Community AIP contains one METSRights section, which describes the permissions different Groups or People have within that Community.

Below is an example of a METSRights sections for a publicly visible Community, which also has an Administrator group. As you'll notice, this content looks very similar to the Collection METSRights section (as described above)

```
<rights:RightsDeclarationMD xmlns:rights="http://cosimo.stanford.edu/sdr/metsrights/" RIGHTSCATEGORY="LICENSED">
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COMMUNITY_hdl:123456789/10_ADMIN</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" COPY="true" DUPLICATE="true" MODIFY="true" DELETE="true"
PRINT="true" OTHER="true" OTHERPERMITTYPE="ADMIN" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="GENERAL PUBLIC">
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
</rights:RightsDeclarationMD>
```