AIP Backup and Restore

1 Background & Overview
   1.1 How does this differ from traditional DSpace Backups? Which Backup route is better?
   1.2 How does this work help DSpace interact with DuraCloud?

2 Makeup and Definition of AIPs
   2.1 AIPs are Archival Information Packages
   2.2 AIP Structure / Format

3 Running the Code
   3.1 Exporting AIPs
      3.1.1 Export Modes & Options
      3.1.2 Exporting just a single AIP
      3.1.3 Exporting AIP Hierarchy
         3.1.3.1 Exporting Entire Site
   3.2 Ingesting / Restoring AIPs
      3.2.1 Ingestion Modes & Options
         3.2.1.1 The difference between "Submit" and "Restore/Replace" modes
      3.2.2 Submitting AIP(s) to create a new object
         3.2.2.1 Submitting a Single AIP
         3.2.2.2 Submitting an AIP Hierarchy
         3.2.2.3 Submitting AIP(s) while skipping any Collection Approval Workflows
      3.2.3 Restoring/Replacing using AIP(s)
         3.2.3.1 Default Restore Mode
         3.2.3.2 Restore, Keep Existing Mode
         3.2.3.3 Force Replace Mode
         3.2.3.4 Restoring Entire Site

4 Additional Packager Options
   4.1 How to use these options

5 Configuration in 'dspace.cfg'
   5.1 AIP Metadata Dissemination Configurations
   5.2 AIP Ingestion Metadata Crosswalk Configurations
   5.3 AIP Ingestion EPerson Configurations
   5.4 AIP Configurations To Improve Ingestion Speed while Validating

6 Common Issues or Error Messages

Background & Overview

AIP Backup & Restore functionality only works with 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.

Additional background information available in the Open Repositories 2010 Presentation entitled Improving DSpace Backups, Restores & Migrations

As of DSpace 1.7, DSpace now can backup and restore all of its contents as a set of AIP Files. This includes all Communities, Collections, Items, Groups and People in the system.

This feature came out of a requirement for DSpace to better integrate with DuraCloud, and other backup storage systems. One of these requirements is to be able to essentially "backup" local DSpace contents into the cloud (as a type of offsite backup), and "restore" those contents at a later time.

Essentially, this means DSpace can export the entire hierarchy (i.e. bitstreams, metadata and relationships between Communities/Collections/Items) into a relatively standard format (a METS-based, AIP format). This entire hierarchy can also be re-imported into DSpace in the same format (essentially a restore of that content in the same or different DSpace installation).

Benefits for the DSpace community:

- Allows one to more easily move entire Communities or Collections between DSpace instances.
- Allows for a potentially more consistent backup of this hierarchy (e.g. to DuraCloud, or just to your own local backup system), rather than relying on synchronizing a backup of your Database (stores metadata/relationships) and assetstore (stores files/bitstreams).
- Provides a way for people to more easily get their data out of DSpace (whatever the purpose may be).
- Provides a relatively standard format for people to migrate entire hierarchies (Communities/Collections) from one DSpace to another (or from another system into DSpace).

How does this differ from traditional DSpace Backups? Which Backup route is better?

Traditionally, it has always been recommended to backup and restore DSpace's database and files (also known as the "assetstore") separately. This is described in more detail in the Storage Layer section of the DSpace System Documentation. The traditional backup and restore route is still a recommended and supported option.
However, the new AIP Backup & Restore option seeks to try and resolve many of the complexities of a traditional backup and restore. The below table details some of the differences between these two valid Backup and Restore options.

<table>
<thead>
<tr>
<th>Supported Backup /Restore Types</th>
<th>Traditional Backup &amp; Restore (Database and Files)</th>
<th>AIP Backup &amp; Restore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can Backup &amp; Restore all DSpace Content easily</td>
<td>Yes (Requires two backups/restores – one for Database and one for Files)</td>
<td>Yes (Though, will not backup/restore items which are not officially &quot;in archive&quot;)</td>
</tr>
<tr>
<td>Can Backup &amp; Restore a Single Community /Collection/Item easily</td>
<td>No (It is possible, but requires a strong understanding of DSpace database structure &amp; folder organization in order to only backup &amp; restore metadata/files belonging to that single object)</td>
<td>Yes</td>
</tr>
<tr>
<td>Backups can be used to move one or more Community/Collection/Items to another DSpace system easily.</td>
<td>No (Again, it is possible, but requires a strong understanding of DSpace database structure &amp; folder organization in order to only move metadata/files belonging to that object)</td>
<td>Yes</td>
</tr>
<tr>
<td>Can Backup &amp; Restore Item Versions</td>
<td>Yes (Requires two backups/restores – one for Database and one for Files)</td>
<td>No (Currently <a href="#ItemLevelVersioning">Item Level Versioning</a> is not fully compatible with AIP Backup &amp; Restore. AIP Backup &amp; Restore can only backup/restore the latest version of an Item)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Object Types During Backup &amp; Restore</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports backup/restore of all Communities/Collections /Items (including metadata, files, logos, etc.)</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports backup/restore of all People/Groups /Permissions</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports backup/restore of all Collection-specific Item Templates</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports backup/restore of all Collection Harvesting settings (only for Collections which pull in all Items via OAI-PMH or OAI-ORE)</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports backup/restore of all Withdrawn (but not deleted) Items</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports backup/restore of Item Mappings between Collections</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports backup/restore of all in-process, uncompleted Submissions (or those currently in an approval workflow)</td>
<td>Yes (During restore, the AIP Ingester may throw a false &quot;Could not find a parent DSpaceObject&quot; error (see <a href="#CommonIssuesOrErrorMessages">Common Issues or Error Messages</a>), if it tries to restore an Item Mapping to a Collection that it hasn't yet restored. But this error can be safely bypassed using the 'skipIfParentMissing' flag (see <a href="#AdditionalPackagerOptions">Additional Packager Options</a> for more details).</td>
</tr>
<tr>
<td>Supports backup/restore of Items using custom Metadata Schemas &amp; Fields</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports backup/restore of all local DSpace Configurations and Customizations</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports backup/restore of all your entire DSpace directory as part of backing up your files</td>
<td>Not by default (unless your also backup parts of your DSpace directory – note, you wouldn't need to backup the '[dspace]/assetstore' folder again, as those files are already included in AIPs)</td>
</tr>
</tbody>
</table>
Based on your local institutions needs, you will want to choose the backup & restore process which is most appropriate to you. You may also find it beneficial to use both types of backups on different time schedules, in order to keep to a minimum the likelihood of losing your DSpace installation settings or its contents. For example, you may choose to perform a Traditional Backup once per week (to backup your local system configurations and customizations) and an AIP Backup on a daily basis. Alternatively, you may choose to perform daily Traditional Backups and only use the AIP Backup as a "permanent archives" option (perhaps performed on a weekly or monthly basis).

Don’t Forget to Backup your Configurations and Customizations

If you choose to use the AIP Backup and Restore option, do not forget to also backup your local DSpace configurations and customizations. Depending on how you manage your own local DSpace, these configurations and customizations are likely in one or more of the following locations:

- `[/dspace]` - The DSpace installation directory
- `[/dspace-source]` - The DSpace source directory

How does this work help DSpace interact with DuraCloud?

This work is entirely about exporting DSpace content objects to a location on a local filesystem. So, this work doesn't interact solely with DuraCloud, and could be used by any backup storage system to backup your DSpace contents.

In the initial DuraCloud work, the DuraCloud team is working on a way to "synchronize" DuraCloud with a local file folder. So, DuraCloud can be configured to "watch" a given folder and automatically replicate its contents into the cloud.

Therefore, moving content from DSpace to DuraCloud would currently be a two-step process:

1. First, export AIPs describing that content from DSpace to a filesystem folder
2. Second, enable DuraCloud to watch that same filesystem folder and replicate it into the cloud.

Similarly, moving content from DuraCloud back into DSpace would also be a two-step process:

1. First, you'd tell DuraCloud to replicate the AIPs from the cloud to a folder on your file system
2. Second, you'd ingest those AIPs back into DSpace

(These backup/restore processes may change as we go forward and investigate more use cases. This is just the initial plan.)

Makeup and Definition of AIPs

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).

AIP Structure / Format

Generally speaking, an AIP is an Zip file containing a METS manifest and all related content bitstreams.

For more specific details of AIP format / structure, along with examples, please see DSpace AIP Format.

Running the Code
Exporting AIPs

Export Modes & Options

All AIP Exports are done by using the Dissemination Mode (-d option) of the packager command.

There are two types of AIP Dissemination you can perform:

- **Single AIP** (default, using -i option) - Exports just an AIP describing a single DSpace object. So, if you ran it in this default mode for a Collection, you’d just end up with a single Collection AIP (which would not include AIPs for all its child Items).
- **Hierarchy of AIPs** (using the -d --all or -d -a option) - Exports the requested AIP describing an object, plus the AIP for all child objects.

Some examples follow:

- For a Site - this would export all Communities, Collections & Items within the site into AIP files (in a provided directory)
- For a Community - this would export that Community and all SubCommunities, Collections and Items into AIP files (in a provided directory)
- For a Collection - this would export that Collection and all contained Items into AIP files (in a provided directory)
- For an Item – this just exports the Item into an AIP as normal (as it already contains its Bitstreams/Bundles by default)

Exporting just a single AIP

To export in single AIP mode (default), use this "packager" command template:

```
[dspace]/bin/dspace packager -d -t AIP -e <person> -i <handle> <file-path>
```

For example:

```
[dspace]/bin/dspace packager -d -t AIP -e admin@myu.edu -i 4321/4567 aip4567.zip
```

The above code will export the object of the given handle (4321/4567) into an AIP file named "aip4567.zip". This will not include any child objects for Communities or Collections.

Exporting AIP Hierarchy

To export an AIP hierarchy, use the -a (or --all) package parameter.

For example, use this 'packager' command template:

```
[dspace]/bin/dspace packager -d -a -t AIP -e <person> -i <handle> <file-path>
```

For example:

```
[dspace]/bin/dspace packager -d -a -t AIP -e admin@myu.edu -i 4321/4567 aip4567.zip
```

The above code will export the object of the given handle (4321/4567) into an AIP file named "aip4567.zip". In addition it would export all children objects to the same directory as the "aip4567.zip" file. The child AIP files are all named using the following format:

- **File Name Format**: `<Obj-Type>@<Handle-with-dashes>.zip`
- e.g. COMMUNITY@123456789-1.zip, COLLECTION@123456789-2.zip, ITEM@123456789-200.zip
- This general file naming convention ensures that you can easily locate an object to restore by its name (assuming you know its Object Type and Handle).
- Alternatively, if object doesn't have a Handle, it uses this File Name Format: `<Obj-Type>@internal-id-<DSpace-ID>.zip` (e.g. ITEM@internal-id-234.zip)

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.

Exporting Entire Site

To export an entire DSpace Site, pass the packager the Handle `<site-handle-prefix>/0`. For example, if your site prefix is "4321", you'd run a command similar to the following:

```
[dspace]/bin/dspace packager -d -a -t AIP -e admin@myu.edu -i 4321/0 sitewide-aip.zip
```

Again, this would export the DSpace Site AIP into the file "sitewide-aip.zip", and export AIPs for all Communities, Collections and Items into the same directory as the Site AIP.
Ingesting / Restoring AIPs

Ingestion Modes & Options

Ingestion of AIPs is a bit more complex than Dissemination, as there are several different "modes" available:

1. **Submit/Ingest Mode** (\(-s\) option, default) – submit AIP(s) to DSpace in order to create a new object(s) (i.e. AIP is treated like a SIP – Submission Information Package)
2. **Restore Mode** (\(-r\) option) – restore pre-existing object(s) in DSpace based on AIP(s). This also attempts to restore all handles and relationships (parent/child objects). This is a specialized type of "submit", where the object is created with a known Handle and known relationships.
3. **Replace Mode** (\(-r\) \(-f\) option) – replace existing object(s) in DSpace based on AIP(s). This also attempts to restore all handles and relationships (parent/child objects). This is a specialized type of "restore" where the contents of existing object(s) is replaced by the contents in the AIP(s). By default, if a normal "restore" finds the object already exists, it will back out (i.e. rollback all changes) and report which object already exists.

Again, like export, there are two types of AIP Ingestion you can perform (using any of the above modes):

- **Single AIP** (default) - Ingests just an AIP describing a single DSpace object. So, if you ran it in this default mode for a Collection AIP, you'd just create a DSpace Collection from the AIP (but not ingest any of its child objects)
- **Hierarchy of AIPs** (by including the \(--all\) or \(-a\) option after the mode) - Ingests the requested AIP describing an object, plus the AIP for all child objects. Some examples follow:
  - For a Site - this would ingest all Communities, Collections & Items based on the located AIP files
  - For a Community - this would ingest that Community and all SubCommunities, Collections and Items based on the located AIP files
  - For a Collection - this would ingest that Collection and all contained Items based on the located AIP files
  - For an Item – this just ingest the Item (including all Bitstreams & Bundles) based on the AIP file.

The difference between "Submit" and "Restore/Replace" modes

It's worth understanding the primary differences between a Submission (specified by \(-s\) parameter) and a Restore (specified by \(-r\) parameter).

- **Submission Mode** (\(-s\) mode) - creates a new object (AIP is treated like a SIP)
  - By default, a new Handle is always assigned
  - However, you can force it to use the handle specified in the AIP by specifying \(-o\ ignoreHandle=false\) as one of your parameters
  - By default, a new Parent object must be specified (using the \(-p\) parameter). This is the location where the new object will be created.
  - However, you can force it to use the parent object specified in the AIP by specifying \(-o\ ignoreParent=false\) as one of your parameters
  - By default, will respect a Collection's Workflow process when you submit an Item to a Collection
  - However, you can specifically skip any workflow approval processes by specifying \(-w\) parameter.
  - **Always** adds a new Deposit License to Items
  - **Always** adds new DSpace System metadata to Items (includes new "dc.date.accessioned", "dc.date.available", "dc.date.issued" and "dc.description.provenance" entries)
  - **WARNING:** Submission mode may not be able to maintain Item Mappings between Collections. Because these mappings are recorded via the Collection Handles, mappings may be restored improperly if the Collection handle has changed when moving content from one DSpace instance to another.

- **Restore / Replace Mode** (\(-r\) mode) - restores a previously existing object (as if from a backup)
  - By default, the Handle specified in the AIP is restored
  - However, for restores, you can force a new handle to be generated by specifying \(-o\ ignoreHandle=true\) as one of your parameters. (NOTE: Doesn't work for replace mode as the new object always retains the handle of the replaced object)
  - Although a Restore/Replace does restore Handles, it will not necessarily restore the same internal IDs in your Database.
  - By default, the object is restored under the Parent specified in the AIP
  - However, for restores, you can force it to restore under a different parent object by using the \(-p\) parameter. (NOTE: Doesn't work for replace mode, as the new object always retains the parent of the replaced object)
  - **Always** skips any Collection workflow approval processes when restoring/replacing an Item in a Collection
  - **Never** adds a new Deposit License to Items (rather it restores the previous deposit license, as long as it is stored in the AIP)
  - **Never** adds new DSpace System metadata to Items (rather it just restores the metadata as specified in the AIP)

**Changing Submission/Restore Behavior**

It is possible to change some of the default behaviors of both the Submission and Restore/Replace Modes. Please see the Additional Packager Options section below for a listing of command-line options that allow you to override some of the default settings described above.

**Submitting AIP(s) to create a new object**

The Submission mode (\(-s\)) always creates a new object with a newly assigned handle. In addition by default it respects all existing Collection approval workflows (so items may require approval unless the workflow is skipped by using the \(-w\) option). For information about how the "Submission Mode" differs from the "Replace / Restore mode", see The difference between "Submit" and "Restore/Replace" modes above.
To ingest a single AIP and create a new DSpace object under a parent of your choice, specify the -p (or --parent) package parameter to the command. Also, note that you are running the packager in -s (submit) mode.

**NOTE:** This only ingests the single AIP specified. It does not ingest all children objects.

```
[dspace]/bin/dspace packager -s -t AIP -e <eperson> -p <parent-handle> <file-path>
```

If you leave out the -p parameter, the AIP package ingester will attempt to install the AIP under the same parent it had before. As you are also specifying the -s (submit) parameter, the packager will assume you want a new Handle to be assigned (as you are effectively specifying that you are submitting a new object). If you want the object to retain the Handle specified in the AIP, you can specify the -o ignoreHandle=false option to force the packager to ignore the Handle specified in the AIP.

### Submitting an AIP Hierarchy

To ingest an AIP hierarchy from a directory of AIPs, use the -a (or --all) package parameter.

For example, use this 'packager' command template:

```
[dspace]/bin/dspace packager -s -a -t AIP -e <eperson> -p <parent-handle> <file-path>
```

for example:

```
[dspace]/bin/dspace packager -s -a -t AIP -e admin@myu.edu -p 4321/12 aip4567.zip
```

The above command will ingest the package named "aip4567.zip" as a child of the specified Parent Object (handle="4321/12"). The resulting object is assigned a new Handle (since -s is specified). In addition, any child AIPs referenced by "aip4567.zip" are also recursively ingested (a new Handle is also assigned for each child AIP).

Another example – Ingesting a Top-Level Community (by using the Site Handle, <site-handle-prefix>/0):

```
[dspace]/bin/dspace packager -s -a -t AIP -e admin@myu.edu -p 4321/0 community-aip.zip
```

The above command will ingest the package named "community-aip.zip" as a top-level community (i.e. the specified parent is "4321/0" which is a Site Handle). Again, the resulting object is assigned a new Handle. In addition, any child AIPs referenced by "community-aip.zip" are also recursively ingested (a new Handle is also assigned for each child AIP).

### AIPs treated as SIPs

This option allows you to essentially use a set of AIPs as SIPs (Submission Information Packages). The default settings will create a new DSpace object (with a new handle and a new parent object, if specified) from each AIP.

### May want to skip Collection Approvals Workflows

Please note: If you are submitting a larger amount of content (e.g. multiple Communities/Collections) to your DSpace, you may want to tell the 'packager' command to skip over any existing Collection approval workflows by using the --workflow flag. By default, all Collection approval workflows will be respected. This means if the content you are submitting includes a Collection with an enabled workflow, you may see the following occur:

1. First, the Collection will be created & its workflow enabled
2. Second, each Item belonging to that Collection will be created & placed into the workflow approval process

Therefore, if this content has already received some level of approval, you may want to submit it using the --workflow flag, which will skip any workflow approval processes. For more information, see Submitting AIP(s) while skipping any Collection Approval Workflows.
Item Mappings may not be maintained when submitting an AIP hierarchy

When an item is mapped to one or more Collections, this mapping is recorded in the AIP using the mapped Collection's handle. Unfortunately, since the submission mode (-s) assigns **new handles** to all objects in the hierarchy, this may mean that the mapped Collection's handle will have changed (or even that a different Collection will be available at the original mapped Collection's handle). DSpace does not have a way to uniquely identify Collections other than by handle, which means that item mappings are only able to be retained when the Collection handle is also retained.

If you encounter this issue, there are a few possible workarounds:

1. Use the restore/replace mode (-r) instead, as it will retain existing Collection Handles. Unfortunately though, this may not work if the content is being moved from a Test DSpace to a Production DSpace, as these existing handles may not be valid.
2. OR, use the submission mode with the "--o ignoreHandle=false". This will also retain existing Collection Handles. Unfortunately though, this may not work if the content is being moved from a Test DSpace to a Production DSpace, as these existing handles may not be valid.
3. OR, remove all existing Item Mappings and re-export AIPs (without Item Mappings). Then, import the hierarchy into the new DSpace instance (again without Item Mappings). Finally, recreate the necessary Item Mappings using a different tool, e.g. the **Batch Metadata Editing** tool supports bulk editing of Collection memberships/mappings.

Missing Groups or EPeople cannot be created when submitting an individual Community or Collection AIP

Please note, if you are using AIPs to move an entire Community or Collection from one DSpace to another, there is a known issue (see DS-1105) that the new DSpace instance will be unable to (re-)create any DSpace Groups or EPeople which are referenced by a Community or Collection AIP. The reason is that the Community or Collection AIP itself doesn't contain enough information to create those Groups or EPeople (rather than info is stored in the SITE AIP, for usage during Full Site Restores).

However, there are two possible ways to get around this known issue:

- **EITHER**, you can manually recreate all referenced Groups/EPeople in the new DSpace that you are submitting the Community or Collection AIP into.
  - Note that if you are using Groups named with DSpace Database IDs (e.g. COMMUNITY_1_ADMIN, COLLECTION_2_SUBMIT), you may first need to rename those groups to no longer include Database IDs (e.g. MY_SUBMITTERS). The reason is that Database IDs will likely change when you move a Community or Collection to a new DSpace installation.
  - OR, you can temporarily disable the import of Group/EPeople information when submitting the Community or Collection AIP to the new DSpace. This would mean that after you submit the AIP to the new DSpace, you'd have to manually go in and add in any special permissions (as needed). To disable the import of Group/EPeople information, add these settings to your dspace.cfg file, and re-run the submission of the AIP with these settings in place:

  ```
  mets.dspaceAIP.ingest.crosswalk.METSRIGHTS = NIL
  mets.dspaceAIP.ingest.crosswalk.DSPACE-ROLES = NIL
  ```

  - Don't forget to remove these settings after you import your Community or Collection AIP. **Leaving them in place will mean that every time you import an AIP, all of its Group/EPeople/Permissions would be ignored.**

Submitting AIP(s) while skipping any Collection Approval Workflows

By default, the Submission mode (-s) always respects existing Collection approval workflows. So, if a Collection has a workflow, then a newly submitted Item will be placed into that workflow process (rather than immediately appearing in DSpace).

However, if you'd like to skip all workflow approval processes you can use the `--w` flag to do so. For example, the following command will skip any Collection approval workflows and immediately add the Item to a Collection.

```
[dspace]/bin/dspace packager -s --w -t AIP -e <eperson> -p <parent-handle> <file-path>
```

This `--w` flag may also be used when **Submitting an AIP Hierarchy**. For example, if you are migrating one or more Collections/Communities from one DSpace to another, you may choose to submit those AIPs with the `--w` option enabled. This will ensure that, if a Collection has a workflow approval process enabled, all its Items are available immediately rather than being all placed into the workflow approval process.

Restoring/Replacing using AIP(s)

**Restoring** is slightly different than just **submitting**. When restoring, we make every attempt to restore the object as it **used to be** (including its handle, parent object, etc.). For more information about how the "Replace/Restore Mode" differs from the "Submit mode", see The difference between "Submit" and "Restore/Replace" modes above.

There are currently three restore modes:

1. **Default Restore Mode** (-r) = Attempt to restore object (and optionally children). Rollback all changes if any object is found to already exist.
2. **Restore, Keep Existing Mode** (`-r -k`): Attempt to restore object (and optionally children). If an object is found to already exist, skip over it (and all children objects), and continue to restore all other non-existing objects.

3. **Force Replace Mode** (`-r -f`): Restore an object (and optionally children) and **overwrite** any existing objects in DSpace. Therefore, if an object is found to already exist in DSpace, its contents are replaced by the contents of the AIP. **WARNING:** *This mode is potentially dangerous as it will permanently destroy any object contents that do not currently exist in the AIP. You may want to perform a secondary backup, unless you are sure you know what you are doing!*

**Default Restore Mode**

By default, the restore mode (`-r` option) will throw an error and rollback all changes if any object is found to already exist. The user will be informed if which object already exists within their DSpace installation.

**Restore a Single AIP:** Use this ‘packager’ command template to restore a single object from an AIP (not including any child objects):

```
[dspace]/bin/dspace packager -r -t AIP -e <eperson> <AIP-file-path>
```

**Restore a Hierarchy of AIPs:** Use this ‘packager’ command template to restore an object from an AIP along with all child objects (from their AIPs):

```
[dspace]/bin/dspace packager -r -a -t AIP -e <eperson> <AIP-file-path>
```

For example:

```
[dspace]/bin/dspace packager -r -a -t AIP -e admin@myu.edu aip4567.zip
```

Notice that unlike `-s` option (for submission/ingesting), the `-r` option does not require the Parent Object (`-p` option) to be specified if it can be determined from the package itself.

In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). In addition, any child AIPs referenced by "aip4567.zip" are also recursively ingested (the `-a` option specifies to also restore all child AIPs). They are also restored with the Handles & Parent Objects provided with their package. If any object is found to already exist, all changes are rolled back (i.e. nothing is restored to DSpace)

**Highly Recommended to Update Database Sequences after a Large Restore**

In some cases, when you restore a large amount of content to your DSpace, the internal database counts (called "sequences") may get out of sync with the Handles of the content you just restored. As a best practice, it is **highly recommended to always** re-run the "update-sequences.sql" script on your DSpace database after a larger scale restore. This database script should be run while DSpace is stopped (you may either stop Tomcat or just the DSpace webapps). PostgreSQL/Oracle must be running. The script can be found in the following locations for PostgreSQL and Oracle, respectively:

- `[dspace]/etc/postgres/update-sequences.sql`
- `[dspace]/etc/oracle/update-sequences.sql`

**More Information on using Default Restore Mode with Community/Collection AIPs**

- Using the Default Restore Mode without the `-a` option, will only restore the **metadata** for that specific Community or Collection. No child objects will be restored.
- Using the Default Restore Mode with the `-a` option, will only successfully restore a Community or Collection if that object along with any child objects (Sub-Communities, Collections or Items) do not already exist. In other words, if any objects belonging to that Community or Collection already exist in DSpace, the Default Restore Mode will report an error that those object(s) could not be recreated. If you encounter this situation, you will need to perform the restore using either the **Restore, Keep Existing Mode** or the **Force Replace Mode** (depending on whether you want to keep or replace those existing child objects).

**Restore, Keep Existing Mode**

When the "Keep Existing" flag (`-k` option) is specified, the restore will attempt to skip over any objects found to already exist. It will report to the user that the object was found to exist (and was not modified or changed). It will then continue to restore all objects which do not already exist.

One special case to note: If a Collection or Community is found to already exist, its child objects are also skipped over. So, this mode will not auto-restore items to an existing Collection.

**Restore a Hierarchy of AIPs:** Use this ‘packager’ command template to restore an object from an AIP along with all child objects (from their AIPs):

```
[dspace]/bin/dspace packager -r -a -k -t AIP -e <eperson> <AIP-file-path>
```

For example:
In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). In addition, any child AIPs referenced by "aip4567.zip" are also recursively restored (the \(-a\) option specifies to also restore all child AIPs). They are also restored with the Handles & Parent Objects provided with their package. If any object is found to already exist, it is skipped over (child objects are also skipped). All non-existing objects are restored.

**Force Replace Mode**

When the "Force Replace" flag (\(-f\) option) is specified, the restore will overwrite any objects found to already exist in DSpace. In other words, existing content is deleted and then replaced by the contents of the AIP(s).

**May also be useful in some specific restoration scenarios**

This mode may also be used to restore missing objects which refer to existing objects. For example, if you are restoring a missing Collection which had existing Items linked to it, you can use this mode to auto-restore the Collection and update those existing Items so that they again link back to the newly restored Collection.

**Potential for Data Loss**

Because this mode actually destroys existing content in DSpace, it is potentially dangerous and may result in data loss! You may wish to perform a secondary full backup (assetstore files & database) before attempting to replace any existing object(s) in DSpace.

**Replace using a Single AIP:** Use this ‘packager’ command template to replace a single object from an AIP (not including any child objects):

```
[dspace]/bin/dspace packager -r -f -t AIP -e <eperson> <AIP-file-path>
```

**Replace using a Hierarchy of AIPs:** Use this ‘packager’ command template to replace an object from an AIP along with all child objects (from their AIPs):

```
[dspace]/bin/dspace packager -r -a -f -t AIP -e <eperson> <AIP-file-path>
```

For example:

```
[dspace]/bin/dspace packager -r -a -f -t AIP -e admin@myu.edu aip4567.zip
```

In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). In addition, any child AIPs referenced by "aip4567.zip" are also recursively ingested. They are also restored with the Handles & Parent Objects provided with their package. If any object is found to already exist, its contents are replaced by the contents of the appropriate AIP.

If any error occurs, the script attempts to rollback the entire replacement process.

**Restoring Entire Site**

In order to restore an entire Site from a set of AIPs, you must do the following:

1. Install a completely "fresh" version of DSpace by following the Installation instructions in the DSpace Manual
   - At this point, you should have a completely empty, but fully-functional DSpace installation. You will need to create an initial Administrator user in order to perform this restore (as a full-restore can only be performed by a DSpace Administrator).
2. Once DSpace is installed, run the following command to restore all its contents from AIPs

   ```
   [dspace]/bin/dspace packager -r -a -f -t AIP -e <eperson> -i <site-handle-prefix>/0 /full/path/to/your/site-aip.zip
   ```

Please note the following about the above restore command:

- Notice that you are running this command in "Force Replace" mode (\(-r\ -f\)). This is necessary as your empty DSpace install will already include a few default groups (Administrators and Anonymous) and your initial administrative user. You need to replace these groups in order to restore your prior DSpace contents completely.
- `\(<eperson>\)` should be replaced with the Email Address of the initial Administrator (who you created when you reinstalled DSpace).
- `\(<site-handle-prefix>\)` should be replaced with your DSpace site’s assigned Handle Prefix. This is equivalent to the `handle.prefix` setting in your `dspace.cfg`
Highly Recommended to Update Database Sequences after a Large Restore

In some cases, when you restore a large amount of content to your DSpace, the internal database counts (called "sequences") may get out of sync with the Handles of the content you just restored. As a best practice, it is highly recommended to always re-run the "update-sequences.sql" script on your DSpace database after a larger scale restore. This database script should be run while DSpace is stopped (you may either stop Tomcat or just the DSpace webapps). PostgreSQL/Oracle must be running. The script can be found in the following locations for PostgreSQL and Oracle, respectively:

- PostgreSQL: `/dspace/etc/postgres/update-sequences.sql`
- Oracle: `/dspace/etc/oracle/update-sequences.sql`

Additional Package Options

In addition to the various "modes" settings described under "Running the Code" above, the AIP Packager supports the following packager options. These options allow you to better tweak how your AIPs are processed (especially during ingests/restores/replaces).

<table>
<thead>
<tr>
<th>Option</th>
<th>Ingest or Export</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>createMetadataFields</td>
<td>ingest-only</td>
<td>true</td>
<td>Tells the AIP ingester to automatically create any metadata fields which are found to be missing from the DSpace Metadata Registry. If 'true', this means as each AIP is ingested, new fields may be added to the DSpace Metadata Registry if they don't already exist. When 'false', an AIP ingest will fail if it encounters a metadata field that doesn't exist in the DSpace Metadata Registry. (NOTE: This will not create missing DSpace Metadata Schemas. If a schema is found to be missing, the ingest will always fail.)</td>
</tr>
</tbody>
</table>
| filterBundles           | export-only      | defaults to exporting all Bundles | This option can be used to limit the Bundles which are exported to AIPs for each DSpace Item. By default, all file Bundles will be exported into Item AIPs. You could use this option to limit the size of AIPs by only exporting certain Bundles. WARNING: any bundles not included in AIPs will obviously be unable to be restored. This option can be run in two ways:
  - Exclude Bundles: By default, you can provide a comma-separated list of bundles to be excluded from AIPs (e.g. "TEXT, THUMBNAIL")
  - Include Bundles: If you prepend the list with the "+" symbol, then the list specifies the bundles to be included in AIPs (e.g. "+ORIGINAL,LICENSE") would only include those two bundles. This second option is identical to using the "includeBundles" option described below.
  (NOTE: If you choose to no longer export LICENSE or CC_LICENSE bundles, you will also need to disable the License Dissemination Crosswalks in the `aip.disseminate.rightsMD` configuration for the changes to take affect) |
<p>| IgnoreHandle            | ingest-only      | Restore /Replace Mode defaults to 'false', Submit Mode defaults to 'true' | If 'true', the AIP ingester will ignore any Handle specified in the AIP itself, and instead create a new Handle during the ingest process (this is the default when running in Submit mode, using the --s flag). If 'false', the AIP ingester attempts to restore the Handles specified in the AIP (this is the default when running in Restore/replace mode, using the --r flag). |
| IgnoreParents           | ingest-only      | Restore /Replace Mode defaults to 'false', Submit Mode defaults to 'true' | If 'true', the AIP ingester will ignore any Parent object specified in the AIP itself, and instead ingest under a new Parent object (this is the default when running in Submit mode, using the --p flag). The new Parent object must be specified via the --p flag (run <code>dspace packager -h</code> for more help). If 'false', the AIP ingester attempts to restore the object directly under its old Parent (this is the default when running in Restore/replace mode, using the --r flag). |
| includeBundles          | export-only      | defaults to &quot;all&quot; | This option can be used to limit the Bundles which are exported to AIPs for each DSpace Item. By default, all file Bundles will be exported into Item AIPs. You could use this option to limit the size of AIPs by only exporting certain Bundles. WARNING: any bundles not included in AIPs will obviously be unable to be restored. This option expects a comma separated list of bundle names (e.g. &quot;ORIGINAL,LICENSE,CC_LICENSE,METADATA&quot;). (NOTE: If you choose to no longer export LICENSE or CC_LICENSE bundles, you will also need to disable the License Dissemination Crosswalks in the <code>aip.disseminate.rightsMD</code> configuration for the changes to take affect) |
| manifestOnly            | both import and export | false | If 'true', the AIP Disseminator will only import/export a METS Manifest XML file (i.e. result will be an unzipped 'mets.xml' file), instead of a full AIP. This METS Manifest contains URI references to all content files, but does not contain any content files. This option is experimental and is meant for debugging purposes only. It should never be set to 'true' if you want to be able to restore content files. Again, please note that when you use this option, the final result will be an XML file, NOT the normal ZIP-based AIP format. |
| export-only             |                   | false | If 'true' (and the 'DSPACE-ROLES' crosswalk is enabled, see #AIP Metadata Dissemination Configurations), then the AIP Disseminator will export user password hashes (i.e. encrypted passwords) into Site AIP's METS Manifest. This would allow you to... |</p>
<table>
<thead>
<tr>
<th>Setting</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>skipIfParentMissing</td>
<td>boolean</td>
<td>If 'true', ingestion will skip over any &quot;Could not find a parent DSpaceObject&quot; errors that are encountered during the ingestion process. Otherwise, these errors will be logged as warning messages in your DSpace log file. If you are performing a full site restore (or a restore of a larger Community/Collection hierarchy), you may encounter these errors if you have a larger number of item mappings between Collections (i.e. Items which are mapped into several collections at once). When you are performing a recursive ingest, skipping these errors should not cause any problems. Once the missing parent object is ingested it will automatically restore the Item mapping that caused the error. For more information on this &quot;Could not find a parent DSpaceObject&quot; error see Common Issues or Error Messages.</td>
</tr>
</tbody>
</table>
| unauthORIZED                 | boolean       | If 'skip', the AIP Disseminator will skip over any unauthorized Bundle or Bitstream encountered (i.e. it will not be added to the AIP). If 'zero', the AIP Disseminator will add a Zero-length "placeholder" file to the AIP when it encounters an unauthorized Bitstream. If unspecified (the default value), the AIP Disseminator will throw an error if an unauthorized Bundle or Bitstream is encountered. If you are programmatically calling the org.dspace.content.packager.DSpaceAIPIngester class, you can specify these options via the org.dspace.content.packager.PackageParameters class. As a basic example:

```java
PackageParameters params = new PackageParameters;
params.addProperty("createMetadataFields", "false");
params.addProperty("ignoreParent", "false");
```

For example:

```
[dspace]/bin/dspace packager -r -a -t AIP -o ignoreParent=false -o createMetadataFields=false -e admin@myu.edu aip4567.zip
```

How to use these options

These options can be passed in two main ways:

From the Command Line

From the command-line, you can add the option to your command by using the --option parameter.

```
[dspace]/bin/dspace packager -r -a -t AIP -o [option1]=value -o [option2]=value -e admin@myu.edu aip4567.zip
```

For example:

```
[dspace]/bin/dspace packager -r -a -t AIP -o ignoreParent=false -o createMetadataFields=false -e admin@myu.edu aip4567.zip
```

Via the Java API call

If you are programmatically calling the org.dspace.content.packager.DSpaceAIPIngester from your own custom script, you can specify these options via the org.dspace.content.packager.PackageParameters class.

As a basic example:

```java
PackageParameters params = new PackageParameters;
params.addProperty("createMetadataFields", "false");
params.addProperty("ignoreParent", "true");
```

Configuration in 'dspace.cfg'

The following new configurations relate to AIPs:

### AIP Metadata Dissemination Configurations

The following configurations allow you to specify what metadata is stored within each METS-based AIP. In 'dspace.cfg', the general format for each of these settings is:

- `<setting> = <mdType>:<DSpace-crosswalk-name> [, ...]`
  - `<setting>` is the setting name (see below for the full list of valid settings)
  - `<mdType>` is optional. It allows you to specify the value of the @MDTYPE or @OTHERMDTYPE attribute in the corresponding METS element.
  - `<DSpace-crosswalk-name>` is required. It specifies the name of the DSpace Crosswalk which should be used to generate this metadata.
  - Zero or more `<label-for-METS>:<DSpace-crosswalk-name>` may be specified for each setting.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| aip.disseminate                |               | The default setting is 'null'. Each AIP will be disseminated to DSpace. This option affects the export of AIPs to DSpace only. For example:

```xml
<setting> = <mdType>:<DSpace-crosswalk-name> [, ...]
```

These settings are specified in the AIP metadata file (METS), and can be set on a per-setting basis. For example:

```xml
<setting> = <mdType>:<DSpace-crosswalk-name> [label-for-METS]:<DSpace-crosswalk-name> [, ...]
```

You can specify these settings in the 'dspace.cfg' file, or you can specify them programmatically using the `org.dspace.content.packager.DSpaceAIPIngester` class. For example:

```java
params.addProperty("aip.disseminate", "true" "false" "true" "false" "false")
```

Exporting AIP Metadata

Settings related to AIP Metadata Dissemination Configurations in 'dspace.cfg' allow you to specify what metadata is stored within each METS-based AIP. In 'dspace.cfg', the general format for each of these settings is:

- `<setting> = <mdType>:<DSpace-crosswalk-name> [, ...]`
  - `<setting>` is the setting name (see below for the full list of valid settings)
  - `<mdType>` is optional. It allows you to specify the value of the @MDTYPE or @OTHERMDTYPE attribute in the corresponding METS element.
  - `<DSpace-crosswalk-name>` is required. It specifies the name of the DSpace Crosswalk which should be used to generate this metadata.
  - Zero or more `<label-for-METS>:<DSpace-crosswalk-name>` may be specified for each setting.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| aip.disseminate                |               | The default setting is 'null'. Each AIP will be disseminated to DSpace. This option affects the export of AIPs to DSpace only. For example:

```xml
<setting> = <mdType>:<DSpace-crosswalk-name> [, ...]
```

These settings are specified in the AIP metadata file (METS), and can be set on a per-setting basis. For example:

```xml
<setting> = <mdType>:<DSpace-crosswalk-name> [label-for-METS]:<DSpace-crosswalk-name> [, ...]
```

You can specify these settings in the 'dspace.cfg' file, or you can specify them programmatically using the `org.dspace.content.packager.DSpaceAIPIngester` class. For example:

```java
params.addProperty("aip.disseminate", "true" "false" "true" "false" "false")
```

Exporting AIP Metadata

Settings related to AIP Metadata Dissemination Configurations in 'dspace.cfg' allow you to specify what metadata is stored within each METS-based AIP. In 'dspace.cfg', the general format for each of these settings is:

- `<setting> = <mdType>:<DSpace-crosswalk-name> [, ...]`
  - `<setting>` is the setting name (see below for the full list of valid settings)
  - `<mdType>` is optional. It allows you to specify the value of the @MDTYPE or @OTHERMDTYPE attribute in the corresponding METS element.
  - `<DSpace-crosswalk-name>` is required. It specifies the name of the DSpace Crosswalk which should be used to generate this metadata.
  - Zero or more `<label-for-METS>:<DSpace-crosswalk-name>` may be specified for each setting.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| aip.disseminate                |               | The default setting is 'null'. Each AIP will be disseminated to DSpace. This option affects the export of AIPs to DSpace only. For example:

```xml
<setting> = <mdType>:<DSpace-crosswalk-name> [, ...]
```

These settings are specified in the AIP metadata file (METS), and can be set on a per-setting basis. For example:

```xml
<setting> = <mdType>:<DSpace-crosswalk-name> [label-for-METS]:<DSpace-crosswalk-name> [, ...]
```

You can specify these settings in the 'dspace.cfg' file, or you can specify them programmatically using the `org.dspace.content.packager.DSpaceAIPIngester` class. For example:

```java
params.addProperty("aip.disseminate", "true" "false" "true" "false" "false")
```
AIP Metadata 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.

The default settings in 'dspace.cfg' are:

- \texttt{aip.disseminate.techMD} - Lists the DSpace Crosswalks (by name) which should be called to populate the \texttt{<techMD>} section of the METS file within the AIP (Default: PREMIS, DSPACE-ROLES)
  - The PREMIS crosswalk generates PREMIS metadata for the object specified by the AIP
  - The DSPACE-ROLES crosswalk exports DSpace Group / EPerson information into AIPs in a DSpace-specific XML format. Using this crosswalk means that AIPs can be used to recreated Groups & People within the system. (NOTE: The DSPACE-ROLES crosswalk should be used alongside the METSRights crosswalk if you also wish to restore the permissions that Groups/People have within the System. See below for more info on the METSRights crosswalk.)
- \texttt{aip.disseminate.sourceMD} - Lists the DSpace Crosswalks (by name) which should be called to populate the \texttt{<sourceMD>} section of the METS file within the AIP (Default: None)
- \texttt{aip.disseminate.rightsMD} - Lists the DSpace Crosswalks (by name) which should be called to populate the \texttt{<rightsMD>} section of the METS file within the AIP (Default: None)
- \texttt{aip.disseminate.digiprovMD} - Lists the DSpace Crosswalks (by name) which should be called to populate the \texttt{<digiprovMD>} section of the METS file within the AIP (Default: None)
- \texttt{aip.disseminate.dmd} - Lists the DSpace Crosswalks (by name) which should be called to populate the \texttt{<dmdSec>} section of the METS file within the AIP (Default: None)

The following configurations allow you to specify what DSpace Crosswalks are used during the ingestion/restoration of AIPs. These configurations also allow you to ignore areas of the METS file (in the AIP) if you do not want that area to be restored.

In \texttt{dspace.cfg}, the general format for each of these settings is:

\begin{verbatim}
mets.dspaceAIP.ingest.crosswalk.<mdType> = <DSpace-crosswalk-name>
\end{verbatim}

- \texttt{<mdType>} is the type of metadata as specified in the METS file. This corresponds to the value of the \texttt{@MDTYPE} attribute (of that metadata section in the METS). When the \texttt{@MDTYPE} attribute is "OTHER", then the \texttt{<mdType>} corresponds to the \texttt{@OTHERMDTYPE} attribute value.
- \texttt{<DSpace-crosswalk-name>} specifies the name of the DSpace Crosswalk which should be used to ingest this metadata into DSpace. You can specify the "NULLSTREAM" crosswalk if you specifically want this metadata to be ignored (and skipped over during ingestion).

By default, the settings in \texttt{dspace.cfg} are:

\begin{verbatim}
mets.dspaceAIP.ingest.crosswalk.DSpaceDepositLicense = NULLSTREAM
mets.dspaceAIP.ingest.crosswalk.CreativeCommonsRDF = NULLSTREAM
mets.dspaceAIP.ingest.crosswalk.CreativeCommonsText = NULLSTREAM
\end{verbatim}

The above settings tell the ingester to ignore any metadata sections which reference DSpace Deposit Licenses or Creative Commons Licenses. These metadata sections can be safely ignored as long as the "LICENSE" and "CC_LICENSE" bundles are included in AIPs (which is the default setting). As the Licenses are included in those Bundles, they will already be restored when restoring the bundle contents.

More Info on Default Crosswalks used

If unspecified in the above settings, the AIP ingester will automatically use the Crosswalk which is named the same as the \texttt{@MDTYPE} or \texttt{@OTHERMDTYPE} attribute for the metadata section. For example, a metadata section with an \texttt{@MDTYPE}="PREMIS" will be processed by the DSpace Crosswalk named "PREMIS".

AIP Ingestion Metadata Crosswalk Configurations

The following configurations allow you to specify what DSpace Crosswalks are used during the ingestion/restoration of AIPs. These configurations also allow you to ignore areas of the METS file (in the AIP) if you do not want that area to be restored.

\begin{verbatim}
mets.dspaceAIP.ingest.crosswalk.<mdType> = <DSpace-crosswalk-name>
\end{verbatim}

- \texttt{<mdType>} is the type of metadata as specified in the METS file. This corresponds to the value of the \texttt{@MDTYPE} attribute (of that metadata section in the METS). When the \texttt{@MDTYPE} attribute is "OTHER", then the \texttt{<mdType>} corresponds to the \texttt{@OTHERMDTYPE} attribute value.
- \texttt{<DSpace-crosswalk-name>} specifies the name of the DSpace Crosswalk which should be used to ingest this metadata into DSpace. You can specify the "NULLSTREAM" crosswalk if you specifically want this metadata to be ignored (and skipped over during ingestion).

By default, the settings in \texttt{dspace.cfg} are:

\begin{verbatim}
mets.dspaceAIP.ingest.crosswalk.DSpaceDepositLicense = NULLSTREAM
mets.dspaceAIP.ingest.crosswalk.CreativeCommonsRDF = NULLSTREAM
mets.dspaceAIP.ingest.crosswalk.CreativeCommonsText = NULLSTREAM
\end{verbatim}

The above settings tell the ingester to ignore any metadata sections which reference DSpace Deposit Licenses or Creative Commons Licenses. These metadata sections can be safely ignored as long as the "LICENSE" and "CC_LICENSE" bundles are included in AIPs (which is the default setting). As the Licenses are included in those Bundles, they will already be restored when restoring the bundle contents.

AIP Ingestion EPerson Configurations
AIP Configurations To Improve Ingestion Speed while Validating

It is recommended to validate all AIPs on ingestion (when possible). But validation can be extremely slow, as each validation request first must download all referenced Schema documents from various locations on the web (sometimes as many as 10 schemas may be necessary to download in order to validate a single METS file). To make matters worse, the same schema will be re-downloaded each time it is used (i.e. it is not cached locally). So, if you are validating just 20 METS files which each reference 10 schemas, that results in 200 download requests.

In order to perform validations in a speedy fashion, you can pull down a local copy of all schemas. Validation will then use this local cache, which can sometimes increase the speed up to 10 x.

To use a local cache of XML schemas when validating, use the following settings in 'dspace.cfg'. The general format is:

```
• mets.xsd.<abbreviation> = <namespace> <local-file-name>
• <abbreviation> is a unique abbreviation (of your choice) for this schema
• <namespace> is the Schema namespace
• <local-file-name> the full name of the cached schema file (which should reside in your [dspace]/config/schemas/ directory, by default this directory does not exist – you will need to create it)
```

The default settings are all commented out. But, they provide a full listing of all schemas currently used during validation of AIPs. In order to utilize them, uncomment the settings, download the appropriate schema file, and save it to your [dspace]/config/schemas/ directory (by default this directory does not exist – you will need to create it) using the specified file name:

```
#mets.xsd.mets = http://www.loc.gov/METS/ mets.xsd
#mets.xsd.xlink = http://www.w3.org/1999/xlink xlink.xsd
#mets.xsd.mods = http://www.loc.gov/mods/v3 mods.xsd
#mets.xsd.xml = http://www.w3.org/XML/1998/namespace xml.xsd
#mets.xsd.dcterms = http://purl.org/dc/elements/1.1/ dcterms.xsd
#mets.xsd.dc = http://purl.org/dc/terms/ dcterms.xsd
#mets.xsd.premis = http://www.loc.gov/standards/premis PREMIS.xsd
#mets.xsd.premisObject = http://www.loc.gov/standards/premis PREMIS-Object.xsd
#mets.xsd.premisEvent = http://www.loc.gov/standards/premis PREMIS-Event.xsd
#mets.xsd.premisAgent = http://www.loc.gov/standards/premis PREMIS-Agent.xsd
```

Common Issues or Error Messages

The below table lists common fixes to issues you may encounter when backing up or restoring objects using AIP Backup and Restore.

<table>
<thead>
<tr>
<th>Issue / Error Message</th>
<th>How to Fix this Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ingest/Restore Error: &quot;Group Administrator already exists&quot;</strong></td>
<td>If you receive this problem, you are likely attempting to Restore an Entire Site, but are not running the command in Force Replace Mode (-r -f). Please see the section on Restoring an Entire Site for more details on the flags you should be using.</td>
</tr>
<tr>
<td><strong>Ingest/Restore Error: &quot;Unknown Metadata Schema encountered (mycustomschema)&quot;</strong></td>
<td>If you receive this problem, one or more of your items is using a custom metadata schema which DSpace is currently not aware of (in the example, the schema is named &quot;mycustomschema&quot;). Because DSpace AIPs do not contain enough details to recreate the missing Metadata Schema, you must create it manually via the DSpace Admin UI. Please note that you only need to create the Schema. <strong>You do not need to manually create all the fields belonging to that schema, as DSpace will do that for you as it restores each AIP.</strong> Once the schema is created in DSpace, re-run your restore command. DSpace will automatically re-create all fields belonging to that custom metadata schema as it restores each item that uses that schema.</td>
</tr>
<tr>
<td><strong>Ingest Error: &quot;Could not find a parent DSpaceObject referenced as 'xx/xxx'&quot;</strong></td>
<td>When you encounter this error message it means that an object could not be ingested/restored as it belongs to a parent object which doesn't currently exist in your DSpace instance. During a full restore process, this error can be skipped over and treated as a warning by specifying the 'skipIfParentMissing=true' option (see Additional Packager Options). If you have a large number of items which are mapped to multiple Collections, the AIP Ingester will sometimes attempt to restore an item mapping before the Collection itself has</td>
</tr>
</tbody>
</table>
been restored (thus throwing this error). Luckily, this is not anything to be concerned about. As soon as the Collection is restored, the Item Mapping which caused the error will also be automatically restored. So, if you encounter this error during a full restore, it is safe to bypass this error message using the 'skipIfParentMissing=true' option. All your Item Mappings should still be restored correctly.

Submit Error: PSQLException: ERROR: duplicate key value violates unique constraint "handle_handle_key"

This error means that while submitting one or more AIPs, DSpace encountered a Handle conflict. This is a general error the may occur in DSpace if your Handle sequence has somehow become out-of-date. However, it's easy to fix. Just run the 

```bash
[dspace]/etc/postgres/update-sequences.sql
```

script (or if you are using Oracle, run: ```bash
[dspace]/etc/oracle/update-sequences.sql
```)
