BitstreamFormat Conversion Instructions

<?xml version="1.0" encoding="utf-8"?>
<html>
Procedure to Upgrade a DSpace 1.5 archive to use the 1.6 BitstreamFormat+Renovation code.

Planning

You have to do a little planning before undertaking the conversion. Review the appropriate documentation (e.g. About Data Formats and BitstreamFormat Renovation) and be able to answer the questions below.

Your most important decision is the choice of which external format registries to configure. This choice reflects the purpose of your DSpace archive, whether you are concerned with digital preservation or just storage and dissemination.

Although the new DSpace architecture allows any number of registries to be configured, it really only makes sense to list one primary registry, and the Provisional registry for locally-added (and hopefully temporary) format identifiers not covered by the primary registry.

First, decide how you want data formats to behave in your archive:

- Is your DSpace mainly used to organize and disseminate assets, so the coarse-grained generic formats (almost like MIME-types) are satisfactory for your purposes?
  * If so, choose the backward-compatible DSpace registry.
- Do you plan to apply digital preservation techniques to the contents of your DSpace? Do you want Bitstreams identified with more fine-grained formats to facilitate preservation activities?
  * If so, you probably want the PRONOM registry, or GDFR when it is available.

Here are all the planning questions, in detail:

1. Which external format registry is your primary registry?
2. *DSpace* - the backward-compatible status quo choice.
3. *PRONOM* - an actual external registry, on the path to later upgrade to GDFR
4. Do you also want to configure the Provisional registry? Should be "yes", but populate it sparingly, only for formats that really need special local entries.
5. Which format identification methods do you want to use?
6. *DSpace* (ONLY if using DSpace registry)
7. *Provisional* (ONLY if using Provisional registry)
8. *DROID* (ONLY recommended if using PRONOM registry)

Outline of the Procedure

Here are the minimal steps you will follow to convert your archive to the new BitstreamFormat model. We recommend keeping this list handy and checking off each step as you complete it:

1. Add DSpace Configuration entries for the format registries and format identifiers you have chosen.
2. Run the "cleanup -c" command to check for Bitstreams without readable asset files.
3. Run Phase 0:

   Upgrade15To16 -0

   (Convert DB schema)

4. Populate the Provisional format registry:

   Upgrade15To16 -p <i>file</i>

   and edit results.
5. Run Phase 1:
6. Run Phase 2:
   Upgrade15To16 -2

   (automatic identification of remaining Bitstreams)

7. Generate detailed Report for records:
   Upgrade15To16 -r -v > report.out

   * Analyze report, summarize translations and check for anomalies.

8. Finish with Phase 3:
   Upgrade15To16 -3

   (final database conversion)

The entire process may take several hours, depending on your choice of format registry. We recommend converting a production DSpace system on separate test server first, working with a copy of your archive. It can share a read-only copy of the asset store with another system, since no Bitstreams are actually written or changed.

**Preparation**

The following instructions refer to two filesystem paths, so be sure you know the correct locations for these:

1. The *install directory*,

   \${DSPACE_INSTALL}\n
   , which is typically under your source distribution where it builds the install hierarchy. It contains the ant script

   build.xml

2. Your *runtime (or home) directory*,

   \${DSPACE_HOME}\n
   , which is the value of *

   dspace.dir

   * in the config file.

**Install Software**

- **Download Patch to DSpace 1.5 Source**

- **Shutdown your DSpace archive, i.e. the Java Servlet container / webserver.**

- **Update your install directory (i.e.**

   \${DSPACE_INSTALL}\n
   **to the 1.6 prototype:** Apply patches to the 1.5 source and rebuild or acquire a 1.6 binary distribution.

- **While in the install directory, run**
ant update

to install new code.
• Check that

$$\{\text{DSPACE\_HOME}\}$$

has a

```
config/formats
```

subdirectory; if not, copy it over from the distribution hierarchy. You’ll probably need to do that.
E.g.

```
cp -rp config/formats $$\{\text{DSPACE\_HOME}\}/config
```

• Edit the DSpace Configuration as outlined below:

Configuration

Add these entries to your DSpace Configuration for the case you chose:

If using "DSpace" Registry

1. a. i. Format Registry Configuration

```java
formatRegistry.DSpace.document = $$\{\text{dspace.dir}\}/config/formats/dspace-formats.xml
formatRegistry.DSpace.validate = true
formatRegistry.DSpace.schema = $$\{\text{dspace.dir}\}/config/formats/formats.xsd
formatRegistry.DSpace.defaultSupportLevel = known
formatRegistry.Provisional.validate = true
formatRegistry.Provisional.schema = $$\{\text{dspace.dir}\}/config/formats/formats.xsd
formatRegistry.Provisional.document = $$\{\text{dspace.dir}\}/config/formats/provisional-formats.xml
```

# 2. Set of registries (not a stack)
```
plugin.named.org.dspace.content.format.FormatRegistry = \norg.dspace.content.format.DSpaceFormatRegistry = DSpace, \norg.dspace.content.format.ProvisionalFormatRegistry = Provisional
```

# 3. Format identifier stack - NOTE: TextSubtypeHelper must come last.
```
plugin.sequence.org.dspace.content.format.FormatIdentifier = \norg.dspace.content.format.DSpaceFormatRegistry, \norg.dspace.content.format.TextHeuristicIdentifier, \norg.dspace.content.format.CSSIdentifier, \norg.dspace.content.format.ProvisionalFormatRegistry, \norg.dspace.content.format.TextSubtypeHelper
```

# 4. CSS SAC parser implementation
```
formatIdentifier.CSSIdentifier.parser = org.w3c.flute.parser.Parser
```

# 5. Identifier(s) returned for CSS hits; first one is canonical.
```
formatIdentifier.CSSIdentifier.identifiers = DSpace:CSS
```

# 6. Add this (set to a Provisional format) if your archive contains tar files.
```
#formatIdentifier.TextHeuristicIdentifier.formats.tar = PRONOM:x-fmt/265
```

# 7. Subtypes of Text formats to be promoted if ID’d by filename extension
```
formatIdentifier.TextSubtypeHelper.subtypes = \nDSpace:XML, DSpace:HTML, DSpace:CSS, DSpace:MARC, DSpace:Postscript, \nDSpace:Mathematica, DSpace:LateX, DSpace:TeX, DSpace:SGML, DSpace:RealAudio
```

Configuration when using "PRONOM" Registry

If you configure PRONOM as your primary format registry, you need
the DROID identifier. We also recommend using the Text Heuristic,
CSS, and TextSubtypeHelper identifiers to make up for shortcomings in DROID.
Test Bitstreams and the Asset Store for Consistency

Before executing the BitstreamFormat conversion, you must ensure the Bitstream information in the database and the asset store are in agreement. Otherwise, if the DB refers to a Bitstream with a missing asset file, the conversion process will fail and waste all the time spent on it so far.

We intentionally designed the conversion to fail when an asset is missing or unreadable because this is an unacceptable flaw in the archive, anyway. If DSpace cannot access some Bitstreams, it cannot fulfill its primary mission. It is reasonable to force the administrator to fix such problems.

We provide a testing and repair command, a variation on the

*cleanup*

script, to find all Bitstreams with missing asset files. It repairs them by marking the Bitstream deleted, and logging it. The DSpace administrator should check the log for failures and decide whether to fix or accept each one, since the next cleanup run will wipe out the "deleted" Bitstreams.

Run the command:

$[DSPACE_HOME]/bin/dsrun org.dspace.storage.bitstore.Cleanup -c
If all goes well, it will display:

```
Checking for missing/unreadable files in asset store
Found 0 missing/unreadable files in asset store.
```

The number in the second line will be nonzero if it finds any problems. See the server logs for ERROR records giving details about each Bitstream, e.g.:

```
(bitstream_id column value appears as
id=209)
```

```
2008-01-09 21:31:00,913 ERROR org.dspace.storage.bitstore.BitstreamStorageManager
@ Bitstream id=209 has missing or unreadable asset file... marking it deleted.
```

Note that it takes approximately 30 minutes to run on a fairly slow server with about 160,000 Bitstreams. Runtime is a factor of the number of Bitstreams, not their overall size.

You can also add the `-n` option to the `Cleanup` command to prevent it from marking broken Bitstreams as deleted. This means, if there are any problems, it will NOT fix them, so you must take care of them manually (or run it again). It may be helpful to use the `-n` option the first time you run this command, so you can check whether any of the Bitstreams reported belong to Items.

**Phase 0: Initial Database Conversion**

This phase modifies your database to the new 1.6 schema, while preserving old data to help in the conversion.

**IMPORTANT:** Ensure there is no interactive access to the archive while you perform these steps, since any other DSpace process accessing Bitstreams may receive incorrect data. Also, another process making concurrent changes to the RDBMS may corrupt your data model.

For the first phase you must

```
cd
```

to the directory containing your DSpace installation, i.e. from where you run

```
ant update
```
Ensure there is a file in the relative path

```
/etc/database_schema_15-16.sql
```

THERE IS NO WAY TO "UNDO" THIS STEP.

```
• • • BACK UP YOUR DATABASE.** *
• • • BACK UP YOUR DATABASE.** *
• • • BACK UP YOUR DATABASE.** *
```

```bash
cd ${DSPACE_INSTALL}
${DSPACE_HOME}/bin/dsrun org.dspace.administer.Upgrade15To16 -0
```

**NOTE:** You can run the

```
org.dspace.administer.Upgrade15To16
```

with various other options to try a "dry run" first, get verbose or debugging output, generate a report, etc. Run it with the

```
--help
```

option

for details.

At this point, all Bitstreams have an undefined format. The two conversion Phases will assign format values to them.

### Populate the Provisional Registry

The purpose of the **Provisional** registry is to provide a separate place for file format entries added by the local DSpace administrator. It is named "Provisional" to remind you that its entries are supposed to be a temporary measure, until the format can be described in a shared external registry such as PRONOM or the GDFR, or even the DSpace built-in registry.

The "Provisional" registry also gives you a separate, distinct, home for all of your local changes and extensions to the format technical metadata. Even for an archive configured with the old, simple "DSpace" format registry, it is helpful to have your local additions in a separate place so when the "DSpace" registry is modified in updates, the changes won't affect or collide with your Provisional registry.

In the conversion process, you will generate a configuration file for the Provisional registry based on the formats used in your archive. The upgrade process will examine the old format registry and create Provisional entries for any formats it finds there which were not part of the original DSpace complement. You must still check and edit this list to ensure the data is correct and no undesirable formats are included. For example, if you are using the PRONOM registry, then you should exclude any Provisional versions of formats already known to PRONOM.

The content of the Provisional registry is dictated entirely by its configuration file. This has the same XML-based format as the DSpace registry's configuration file so you may consult that as an example, see

```
config/formats/dspace-formats.xml
```

under the runtime directory.
To populate the Provisional registry:

1. Generate an automatic guess at the formats needed:
   
   ```bash
   \${DSPACE_HOME}/bin/dsrun org.dspace.administer.Upgrade15To16 -p new.xml
   ```

2. _Edit configuration file _

   ```
   new.xml
   ```

   , deleting the

   ```
   &lt;entry&gt;
   ```

   elements for any unwanted or unneeded formats. Be careful to preserve the XML format.

3. Move the new file into place:

   ```
   cp new.xml \${DSPACE_HOME}/config/formats/provisional-formats.xml
   ```

The next upgrade phase will automatically check the validity of your configuration file. It will stop if there is a problem.

**Phase 1: Converting Formats in DSpace and Provisional Registries**

This phase converts Bitstreams whose format has a direct analogue in either the DSpace (if configured) or Provisional registry. If those registries are not configured (or the Provisional registry is empty), it does nothing.

You can run Phase 1 more than once; in fact, you'll have to if you change the Provisional registry after the first run. It does no harm to run Phase 1 again, since it does nothing more if the registries have not changed.

Run this command:

```bash
${DSPACE_HOME}/bin/dsrun org.dspace.administer.Upgrade15To16 -1
```  

Typical Phase 1 results (this example took about 4 minutes):

```
Phase One Summary: Bitstream Formats Converted:
Out of 155185 upgradable bitstreams in archive,
Total bitstreams touched = 155185
Had old "Unknown" format = 1021
   of Unknown, touched 33 with UserFormatDescription set.
Translated to DSpace namespace = 0
Translated to Provisional namespace = 13
```

At this point you may make changes to the contents of the Provisional registry, if, for example, fewer Bitstreams were converted than you expected. If you have no changes to make to the Provisional format registry you may proceed to Phase 2.

**Phase 2: Automatic Format Identification**

This phase assigns new BitstreamFormats to all Bitstreams not already converted, by automatically identifying their formats. It will not touch Bitstreams which have already been identified by conversion.

- If you are keeping the old DSpace registry, this only includes Bitstreams such as license files that never had proper formats, so it will be fast.
- If you are converting to the PRONOM registry, most Bitstreams will have to be identified in Phase Two. This may take several hours.

After running Phase 2 you will not be able to run Phase 1 again.
You may wish to test this Phase first by adding the dry run and verbose options

```
-n -v
```

to the command and watching the output for a while:

```
${DSPACE_HOME}/bin/dsrun org.dspace.administer.Upgrade15To16 -2 -n -v
```

Be sure you'll get adequate results when you run Phase 2 for real, because it assigns some format to each as-yet unidentified Bitstream, which means it will ignore them on subsequent runs.

Don't worry too much about getting everything perfect the first time, though. You can always re-identify the formats of specific Bitstreams and groups of them with the BitstreamFormat Workbench utility, even after the conversion is finished.

To invoke Phase 2:

```
${DSPACE_HOME}/bin/dsrun org.dspace.administer.Upgrade15To16 -2
```

Typical phase 2 results: it takes about 60 minutes to process process 64,000 Bitstreams on a fast server, (almost 5 hours for 155,00 Bitstreams on a slow server), and the summary looks like:

```
Phase Two Summary: Bitstream Formats Guessed:
Total bitstreams touched = 64226
Automatically identified = 64082
Had old "Unknown" format = 106
 Converted to new "Unknown" = 144
```

You should only run Phase 2 once, since it does not leave any Bitstreams unidentified, and it will not touch a Bitstream which has already been identified.

### Reports and Verification

You should produce a detailed report and save it just before finishing the conversion.

You can generate a report of the state of the conversion, and optionally details about each Bitstream, at any time before Phase 3 has run. Invoke the command:

```
${DSPACE_HOME}/bin/dsrun org.dspace.administer.Upgrade15To16 -r
```

This produces a simple summary report like:

```
SUMMARY Total upgradeable Bitstreams = 64269
SUMMARY Already converted to New BSF = 43
SUMMARY Not yet converted to New BSF = 64226
SUMMARY BS with UserFormatDescription = 29
```

To add the details of how each Bitstream was converted from the old format to its new one, add the verbose ( -v ) option, for example:

```
${DSPACE_HOME}/bin/dsrun org.dspace.administer.Upgrade15To16 -r -v
```
This adds a line for each Bitstream in the following format:

1. BITSTREAM, <BitstreamID>, <Name>, <Old BSF Name>, <New BSF>, <New Confidence>, <New Source>, <User Format Desc.>

   BITSTREAM, 5306, OR-083-78.pdf, Adobe PDF, (NONE), UNIDENTIFIED, "null",
   BITSTREAM, 53459, MIT-EL-88-003WP-19737424.pdf, Adobe PDF, (NONE), UNIDENTIFIED, "null",
   BITSTREAM, 163802, license.txt, License, 7-bit ASCII Text, HEURISTIC, "org.dspace.content.format.TextHeuristicIdentifier",
   ... 

We strongly recommend archiving a copy of the detailed report. Run the reporting command above with the output directed into a file. This gives you a record of the original old-style format of each Bitstream, and the vital information about the initial conversion. You can also analyze the report as shown in the next section to check the accuracy of the conversion.

Analyzing Conversion Reports

Here is how to analyze the report to verify the format conversions make sense.

1. Create a file with the report output, e.g.
   "report.out"

2. Use a simple Unix (Linux) text filter to summarize the conversions by reducing the report to unique combinations of old format, new format, and confidence, with the command:
   \texttt{awk -F, '/^BITSTREAM/ \{print $4,,$5,,$6\}' < report.out | sort -u}

3. Examine this output, which has the old format in the first column, then the name of the new format, and the confidence value.
   
   Consider whether the conversion makes sense or not – e.g.
   Postscript
   to
   PostScript 2.0
   is fine, but
   Adobe PDF
   to
   ASCII Text
   may be the sign of a problem.
   
   If you find an anomaly or problem, use the following procedure to examine it further.

For example, suppose you notice an anomaly and want to take a closer look at the affected Bitstreams – perhaps examine a Bitstream itself to see what format it really seems to be. The anomalous line from the summary report is:

Adobe PDF , Windows Portable Executable , POSITIVE_SPECIFIC

Use \texttt{awk} to pick out the relevant lines from the original report by matching the old BSF name in the fourth column and the new BSF name in the fifth column. It should be sufficient to match against a regular expression representing part of each name rather than match the whole thing, for example:
To print only the bitstream DB ID number, modify the awk script:

```bash
% awk -F, '$4 ~ /Adobe/ && $5 ~ /Executable/ {print $2}' < report.out
5153
```

Now you can examine the Bitstream by visiting it through the DSpace web server, e.g.

```text
http://dspace.myuni.edu/dspace/retrieve/5153
```

If corrections are necessary, use the BitstreamFormat Workbench

(organization)

) to re-guess or manually set the format based on that Bitstream number. You can do this after Phase Three just as well as now, so it works even if you only discover the anomaly long after finishing this conversion.

"Undo" and Starting Over

If you discover you've made a terrible mistake and mis-identified a lot of Bitstreams, you can return to the state just before Phase One and re-do all the steps (populating the Provisional registry, etc.). Execute the following SQL statement in your database environment:

```sql
UPDATE Bitstream SET bitstream_format_id = NULL;
```

Another way to return your system to that same state is to restore your database from the backup you made before Phase 0 (right?) and then re-run Phase 0 and the subsequent steps.

Also see the last section about making repairs after finishing the conversion procedure.

Phase 3: Database Cleanup

ONLY begin this phase when you are satisfied with the results of Phase 2, and all of the Bitstreams in the archive have been converted. This phase removes the remaining DSpace 1.5 BitstreamFormat data structures from the RDBMS since they are no longer needed.

This phase checks that all Bitstreams have been converted and will only run if they have been.

```
${DSPACE_HOME}/bin/dsrun org.dspace.administer.Upgrade15To16 -3
```

Typical output: (after a couple seconds of runtime)

```
Phase 3 finished, done with BitstreamFormat conversion.
```

Maintaining and Repairing BitstreamFormats

If a problem or mistake in the BitstreamFormat conversion only becomes apparent after you finish the conversion process (Phase 3) and thus cannot go back to it, you can still put it right. The BitstreamFormat Workbench tool,
Troubleshooting: Adding Filename Extensions to a PRONOM Format

PRONOM/DROID is currently missing (or has a bug preventing it from matching) some filename extensions, also known as external signatures. For example, though the PRONOM entry for an HTML format includes both the common three-letter (MS-DOS) .htm extension and the conventional .html extension, DROID fails to recognize "*.htm" files.

Here is a temporary workaround procedure you can use until DROID is fixed, and for possible future cases where PRONOM/DROID lacks signatures:

1. Add an entry to the Provisional registry with the desired filename extension.
2. Find the BSF with the PRONOM identifier, and add the Provisional entry you just created as a synonym external identifier.
3. If the format is a subtype of "plain text" (i.e. if the PRONOM format is listed in TextSubtypeHelper), add the new Provisional identifier there too.

Here are detailed instructions for an example that adds the .htm filename extension to PRONOM:fmt/96

Provisional Registry entry

Create an entry for the Provisional registry like the one shown below:

1. name may be anything unique in the registry.
2. description should include an explanation of why it is there.
3. external-identifiers should include its identifier and the PRONOM synonym(s)
4. external-signatures includes all desired filename extensions
Modify BitstreamFormat Registry

1. Go to the administrative GUI for the BitstreamFormat Registry.
2. Edit the BitstreamFormat entry for external identifier
   
   PROVONOM:fmt/96

3. Add the external format entry
   
   Provisional:HTM

to it. (Click Add New)

Modify DSpace Configuration

If the PRONOM format was a subtype of Text, for the purpose of

TextSubtypeHelper

, then you must add the Provisional external identifier to the subtype list too.

Locate the configuration entry

formatIdentifier.TextSubtypeHelper.subtypes

, and add the external identifier

Provisional:HTM

to it.

Testing

Find a Bitstream with the

.htm

filename extension that was not identified correctly before, and retry identifying it, e.g. with the BitstreamFormat Workbench utility.