BitstreamFormat Renovation Use Cases

`needsupdate`
This list of use-case sketches demonstrates how the
BitstreamFormat Renovation
proposal will work, and shows some of the scenarios
the designers had in mind. They are "sketches" of use cases because
they do not have the exhaustive detail and exploration of alternatives
required for a full
use case document.

Ingest

Use case sketches about ingest operations.

Interactive Ingest

When submitting a new Item interactively, the user creates Bitstreams by uploading the contents through a Web browser. This supplies DSpace with a filename and possibly a MIME type, but no other clues to the data format except the contents of the Bitstream itself.

Upon receiving each Bitstream, the ingest service calls on the new automatic format identification service to assign it a `BitstreamFormat`. It also returns a "quality" metric indicating the certainty of the identification.

At this point the UI should display the identified format for confirmation by the user. It can also use the quality to advise the user on whether they need to check the automatic results; e.g. for the very weakest levels of quality. Also offer the user the option of overriding the format choice,
see Interactive Format Selection.

Uploading Logo Images

The UI for creating and modifying Collections and Communities allows new "logo" images to be uploaded. The procedure for these is almost exactly the same as for the contents of Items, except that the UI should also check that the identified format is an acceptable image (i.e. test that its MIME type begins with "image/").

Unattended Ingest

The non-interactive (i.e. batch) ingestion methods benefit especially from a more reliable and accurate automatic identification of data formats.

Package-based Ingest

Each ingested Bitstream (both content *and* metadata) has its data format automatically identified.
The submission information package (SIP)
can potentially deliver three pieces of relevant metadata for each Bitstream:

1. Filename, including the extension that may indicate type.
2. MIME type (not useful for format identification)
3. data format identifier from a known external registry

One source of the *data format identifier* is a
PREMIS *object* element,
which specifies the registry as well as the identifier. If that
format registry is known to the ingesting DSpace archive (i.e.
configured as an external registry by the `BitstreamFormat`
implementation), then a simple lookup will return the exact
`BitstreamFormat` referring to that format and we can
accept that as the correct format, if the source of the package is trusted.

If there is no format identifier from a known format registry, then the automatic format identifier is invoked as in the interactive case.

Low-quality or failed format identifications should result in a warning.

NOTE: There is currently no designated mechanism to collect and deliver warnings during a non-interactive process. Messages can be sent to the Java logging facility, but that collects messages for all DSpace processes running on the server. There ought to be a way for a single application to collect its own warnings, and later deliver them.

The MIME type, if found, *could* be used to identify conflicts and possible mistakes in automatic format identification. Especially if the first word of the MIME type is different between package and identified type (e.g. it sent a "image" but was received as "audio"), then a warning should be recorded.

Mirror, or Custody Transfer of Item between DSpace archives

To move or copy an Item between archives, the source disseminates it as some sort of package, which the target then ingests. Ideally they use an actual Archive Information Package (AIP) so there is no loss of data or metadata when crosswalking to intermediate formats, which is inevitably incomplete.

The operation is successful if the new object is identical in content and behavior to the source object. This implies the `Bitstreams` have precisely equivalent `BitstreamFormat` values.

The copying of a Bitstream follows this sequence of operations:

1. Encode its `BitstreamFormat` as technical metadata in outgoing package, by adding each global, external identifier for the format to the PREMIS metadata in a `formatRegistry` element.
2. Ingester finds `formatRegistry` elements in the PREMIS, and so long as it has access to the same external format registry, it can create the equivalent `BitstreamFormat`.
    - Possibly sanity-check on ingest that all recognized identifiers map to the same `BitstreamFormat`.

This assumes that both the source and target DSpace archives have the same format registries configured. But what if the source has GDFR and PRONOM configured, while the target only has GDFR? The target will still match the GDFR identifier in the PREMIS and assign the correct local `BitstreamFormat` to the Bitstream. However, when it produces an AIP or DIP of the Item, that Bitstream will only have the GDFR format identifier: some data has been lost, because the content model relies on `BitstreamFormat` objects to manage format identifiers. Ingested objects have their formats *normalized* to the local archive's format model, i.e. the collection of external registries it configures.

In practice, this should usually not matter. When our example Item is sent back to its original archive, the Bitstream will get back its original `BitstreamFormat` – because the GDFR format identifier sent with it will get resolved to the same `BitstreamFormat` it had originally.

If two DSpace archives are exchanging a lot of Items, they should be configured with the same data format registries (or at least an overlapping set), so the format technical metadata on their Bitstreams is mutually comprehensible.

ItemImporter

The traditional ItemImporter ingests Items from a local file structure on the DSpace server. The only cues it has available to identify a Bitstream's data format are its filename (i.e. external signature) and the contents of the Bitstream itself. Format identification works the same as the package-based ingester (at least, in cases where the package does not contain explicit format identifiers).

The addition of "quality of identification" results makes it easier for archive administrators to evaluate the success of an import and determine whether to review the automatic format choices.

Dissemination

Here is how the new format design is used in various dissemination tasks:

Interactive

Single Bitstream over HTTP

DSpace's current Web-based user interfaces deliver content by sending the raw Bitstream's data stream to the browser via HTTP. To conform to this protocol, they must describe the data format with a MIME type, in `Content-Type:` header sent as part of the HTTP response message. See also W3C statement on MIME types. HTTP clients such as Web browsers *should* **depend entirely on the MIME type** to render the content correctly In practice, some clients cheat and look at filename extensions as well, but this is irregular and should be unusual. Therefore it is critical for DSpace to apply the correct MIME type on materials disseminated through HTTP.

If the MIME type of a Bitstream is "wrong", i.e. does not match what the browser expects for its format, it will not be rendered correctly. The user will be prompted for instructions if the MIME type is unknown to the browser. Note that the definition of "wrong" is somewhat situational; since MIME Type strings are poorly standardized, there are several valid descriptions of some formats, but commonly-used browsers may only recognize one of them.

Tweaking MIME Type

If Bitstreams are disseminated with a MIME Type that the prevailing browser does not recognize, this can lead to pressure on the DSpace administrator to change the behavior of his application, especially in an academic environment. For example, when a course uses digital objects from DSpace, they must be accessible to the browsers commonly used by the students.

Fortunately, the MIME type is a characteristic of the `BitstreamFormat` object, so changing it there will change the MIME type applied to all Bitstreams of that format. So, to alter the MIME type applied to a class of Bitstreams, the administrator only has to go to the DSpace admin interface and change the appropriate `BitstreamFormat`.

This requires the DSpace administrative GUI to provide an obvious path from a Bitstream's description to its `BitstreamFormat`, and the administrative interface for that format.

Although the MIME Type of a `BitstreamFormat` initially comes from its external format registry entry, it is subject to local override. This means the administrator can edit the local archive's `BitstreamFormat` to alter the MIME type, and the change is *persistent*, so it remains even if the `BitstreamFormat` is updated from its external registry.

Unattended

Package-based Dissemination

The METS package profiles for DSpace SIPs and AIPs call for Bitstream technical metadata in several places:

- MIME Type in the `file` element.
- MIME Type in the PREMIS object section.
- Registry-based format identifiers in PREMIS `formatRegistry` element.

All of them have obvious sources in the `BitstreamFormat` object. The MIME type or external registry identifiers are simply taken from the Bitstream's `BitstreamFormat` when adding technical metadata to the package.

Search

The file format can be the object of a search query. Usually it requires a very coarse-grained view of formats, *e.g.* "image", or "audio". Searchers are typically looking for Items that include, *e.g.*, an image or audio component. The Dublin Core **type** element is supposed to describe the nature of the content as both a media type and purpose or venue i.e. types of text media are distinguished as "Article", "Thesis", "Monograph", etc. If the submitter did not provide a **type** value, perhaps it could be derived from the format types of content Bitstreams. This would only capture the media-type sense of **type**, but perhaps that is better than nothing?

This still raises the question of mapping the fine-grained format definitions we labor to identify so precisely onto the coarse range of values assigned to the DC **type** element. None of the present external format registries include such coarse-grained metadata for formats. The prefix of the MIME type might be made to serve, e.g. `text, image, audio,` *etc.*

Archive Admininistration

Interactive Format Selection

There are many tasks and dialogs in the Web user interface where the user is *offered* the option to select a data format for a Bitstream (note that it is not always necessary to *use* it; *e.g.* when the format has been identified automatically already, the manual selection is only needed if that result was unsatisfactory):

- Confirmation dialog after uploading a Bitstream while submitting an Item.
- Workflow tasks to revise the metadata of a pending Item. Show "quality" of previous format identification as well as result, allow changes.
- Administrative functions to edit the metadata of an archived Item.
- Administrative pages to upload "logo" images when creating or editing Collection and Community objects.

Since the internal collection of {{BitstreamFormat}}s (BSFs) is now just a "cache" for entries in external format registry, it is not sufficient to give the user a choice of existing BSF entries. If they are looking for a format which has not been seen already in the archive, there will not be a BSF for it. To get access to the exhaustive list of data formats, we must offer the user a choice from amongst all of the formats in each of the configured external registries, or at least the most complete and preferred registry.

Listing and Navigating Formats

When presenting a choice of formats to the user, the fundamental problem is that the external format registries have many, many entries – from 500 to thousands. This is too many to put in a simple pulldown menu. The registries each have their own metadata and tools to help users select a format. Rather than force a common interface on all registries, we propose that the DSpace UI defer to the format registry's UI to select a format, or perhaps implement a plugin-style UI dialogue to interface with the registry.

The registry's own search tools are bound to be more effective and powerful
than a generic approach.
PRONOM, for example,
allows searching for formats by the software or vendor that produces them,
which is more understandable to the naive user.

All DSpace requires from the format selection process is a
namespaced external format
identifier, which will either be matched to an existing BSF or ingested
to create a new one. It is a simple matter for the module that
accepts the results of a registry-specific UI to add the DSpace-specific
namespace to the identifier, since the registry is already known.

Editing Format Metadata

Although a `BitstreamFormat` object in the DSpace content
model is created by ingesting an external format description, most of
the format's metadata may then be modified. The modifications act as
persistent
local "overrides" of the remote format data, so they remain even if the
format is re-ingested after its remote source is updated.
Any changes are local to the DSpace archive;
they do *not* get reflected out to the
external format registry. The
modifiable properties include:
*__Name:__ Affects how the format is displayed in the UI.
*__Description:__ Detailed description available through UI.
*__MIME Type:__ Affects how Bitstreams are disseminated through HTTP.
*__Canonical Extension:__ Can be used to generate filenames, might be needed to accomodate broken HTTP user agents and when making up filenames in dissemination packages (DIPs).
*__Support Level__ specifies policy regarding the level of commitment to preserve Bitstreams of this format.

Adding New File Formats

Sometimes it is necessary to add a new file format to the repertoire
of the external format registries (note that new BSFs are added
automatically for any unrecognized external identifier).
When the true format of a Bitstream is not already listed in any of the
configured external registries, it must be added somehow.
The options are, in order of preference:

1. Add a full description of the format to a user-editable external registry, such as the GDFR. It will create a globally unique identifier in its namespace.
    - Include the data to drive automatic format identification tools, e.g. internal signatures.
    - Supply other metadata called for by the registry, such as references to specification documents.
2. Add a format entry to the built-in **Local** registry in your DSpace, make up a locally-unique identifier.
    - References to this format are *not* portable to other DSpaces unless they have the same Local format entry (i.e. it was manually copied over).

The first option is much preferred, since adding a the format description
to a common registry benefits all of its users, and gives you a persistent
format identifier.

Managing Bitstreams

An archive administrator sometimes needs to modify format technical metadata
in the content model to correct mistakes or accomodate changes. Some
possible scenarios:

1. One Bitstream in an Item is discovered to have the wrong format, or none, and must be corrected.
2. Many Bitstreams did not have their format automatically identified in a recent batch import. After fixing the automatic identification, they must be re-identified.
3. A new format description is added to the **PRONOM** registry, deprecating a format in the **Local** registry that was added for expedience.
4. **Change all Bitstreams referring to the old format over to the new one, and delete the*Local** entry.

The administrative UI needs a method to select a collection of Bitstreams
by their `BitstreamFormat`, among other critera. (Note that since
unidentified Bitstreams are set to the *Unknown* format, they are selected
as easily as any other format.) This collection can
then be the subject of other operations, namely:

- Change format of selected Bitstreams to a different BSF.

- Re-try automatic format identification.
    - Must offer confirmation option when done interactively.

Some administrative operations are needed for the {{BitstreamFormat}}s themselves:

- Delete a BSF (provided no Bitstreams refer to it, of course).
- Add or modify the external identifiers mapped to a BSF.
- Edit descriptive and administrative metadata.
- Locate the BSF for a given external identifier.

Assessments and Reports

The new format infrastructure gives the archive administrator much more control over how formats are identified and even where the technical metadata comes from. In order to make intelligent decisions and monitor their outcome, she needs to gather data about the archive, so these reports will be available:

- Histogram of number of Bitstreams referencing each BSF.
- Counts of each format-identification quality for each type of BSF.
- Dump of all BSF table entries.
- Dump of all external identifiers bound to BSFs, organized by registry.

The histogram of BSF usage is especially important since it can be coupled with alerts about obsolete formats to gauge how serious the problem is. It can also show how effectively the format identification works by the frequency of precise format versions versus generic broadly-defined formats. The report of quality per BSF shows that more graphically and can help tune the format identification configuration.

Preservation Tasks

The following digital preservation tasks depend on features of the data format infrastructure:

Format Identification

Virtually all preservation tasks depend on knowing the exact data format of the digital object being preserved, so accurate *format identification* is the cornerstone of DSpace preservation.

We believe it is better to concentrate on making automatic format identification precise, accurate, and efficient, since it is likely to be more reliable than manual format identification. Few end-users understand the importance and subtleties of data format identification, or appreciate the advantage of having thousands of known formats to choose from. In our experience the average submitter, or even the average workflow editor, is not likely to give more than cursory attention to format technical metadata.

Newly-ingested Bitstreams

It is critical to correctly identify the formats of Bitstreams in new submissions at the time of ingestion; once the Item is in the archive, it is not guaranteed to get any more attention even from administrators.

This requirement can be satisfied by a configurable policy and the mechanism for enforcing it. For example, the policy would state the minimum acceptable format identification *quality*, and the consequence for failure. Bitstreams receiving a quality metric below the minimum would result in one of these alternatives:

- Ingestion operation fails.
- Ingested item is held (as in workflow) for administrative checking and approval.
- Warning is logged and sent to ingester, and owner of target Collection.
- No consequences.

Since the range of *quality* includes `NONE`, meaning no format was identified, setting the minimum acceptable quality to `NONE` is another way to allow failures with no consequence.

The proposed format-identification policy is configurable at each Collection and as a default for the entire archive.

Tuning Format Identification

The machinery of automatic format identification is completely configurable, so the administrator of each DSpace instance can adapt it to suit his needs. It is implemented as a sequence plugin, in which the implementations are all called in a configured order. Each plugin may recognize only some formats, and it can also see and leverage the results of previously-called plugins.

Tuning the sequence of format identification plugins lets the archive administrator keep up with new data formats and the constant improvments in the technology of identifying them. As well, each archive has its own requirements of format identification, based on the types of material it ingests and the requirements for its preservation.
To tune format identification:

1. Determine the range of formats that need to be identified, and desired precision
   - E.g. is "XML" adequate for all XML-based formats, or do some need to be identified as e.g. SVG, XHTML, METS..
2. Select which plugin implementations to include and the most advantageous ordering.
3. Test against samples of expected submissions, and revise if necessary.
4. Keep up-to-date on format identification developments:
   - Watch for news and exchange information within the DSpace community.
   - Revise configuration as needed.

Detect Obsolete Formats

When preserving digital objects, it is essential to know when their format is becoming obsolete and thus needs attention. See the AONS II project for an example of an application that does this.

First, you need very fine-grained format identification that discriminates between versions of a family of formats (e.g. PDF). Often, older versions of a format will become unsupportable while the later versions are still viable.

This task also illustrates another advantage of using external data format registries as the archetype of DSpace format definitions: we automatically leverage the work of preservation specialists maintaining and using those external registries, e.g. when they announce obsolete formats.

When a format in an external registry is declared obsolete, the DSpace administrator can easily locate Bitstreams in that format using the same tools as for updating and changing formats.

The archive's actions should also be governed by policy, namely the **support level** property of the `BitstreamFormat`. If the support level is anything less than `SUPPORTED`, then the archive may ignore the obsolecense of that format or just issue a warning to owners of affected resources. Otherwise it is obligated to migrate or otherwise preserve the affected Bitstreams.

Selection of Applications

A primary use of file format technical metadata is to match Bitstreams to the applications and filters that can accept their format. The preservation-tool framework within DSpace helps manage this process by finding tools that support a Bitstream's format.

An application is configured in the framework by listing:

1. The plugin interface it implements (e.g. `MediaFilter`)
2. File formats it accepts, in the form of namespaced external format identifiers.
    - Should include the appropriate format(s) for each external registry in use.
3. Optionally, output format it produces.

If an application is capable of producing multiple output formats, it would be configured as multiple instances, since each instance will also need some sort of parameter to tell it which format to emit.

Matching Format Families and Supertypes

One problem in configuring an application is when it claims to accept all versions of a format, or simply is not specific beyond a generic description like "MS Word documents": how can you translate that to format entries in a registry? To configure it properly in DSpace, it seems you'd have to hunt down all the specific format definitions that fit the broad profile of what it accepts – or else be able to configure a non-specific format, as described here.

Allowing an application to configure its "accepted formats" as generic or non-specific formats has these advantages:

- Much easier and more likely to be done correctly by the DSpace administrator.
- As formats are added to the registry, *e.g.* new version of a format within a family, the configuration will still be correct without any updates.
- Reflects the reality of unspecific input requirements of the application.

Some external format registries, such as GDFR and PRONOM, have a hierarchical type model for formats. They document formats which are subtypes of other "supertypes", or belong to a family of formats headed by a generic format. Each registry has different subtle distinctions in its relationship model, which makes it difficult to create a "normalized" view of it in the DSpace `BitstreamFormat` model. Another reason not to model it in `BitstreamFormat` is that the supertype mentioned in the configuration may not have any entry in the `BitstreamFormat` table, since only formats referenced by Bitstreams are included there.

Rather than attempt a flawed normalization, we will interface to external type hierarchies through the `FormatRegistry` plugin. Since we only need to answer the question, "Is this format acceptable as input to an application that says it accepts Format X?", we can just add a method to ask that question directly of the external registry: Is this format equivalent to or a subtype of X?

Here is an illustrative example:

1. Start with a document identified as PRONOM `"fmt/98"` (HTML 3.2) format.
2. We want to get plain text out of it (e.g. for full-text indexing).
3. There is a filter configured that accepts the PRONOM `"fmt/96"` (generic "HTML")
    - Asking the registry, we discover `"fmt/98"` is a subtype of `"fmt/96"`.
    - The Bitstream is therefore acceptable to that filter, start processing.

Data Format Validation

Validation is a necessarily distinct task from format identification. Not only is it a waste of time to try validating a format when it has not been precisely identified yet, there is also the possibility of false positives. Validators and format identification have different goals, anyway: for example, a PDF validator only has to ensure the document conforms to the specification of the PDF version(s) it validates, while the identifier must accept any arbitrary byte stream without crashing and identify all the formats it knows. A loose validator may not discriminate between different versions of a format, while the identifier must do so. It is valuable to be able to identify formats even if we cannot validate them.

Validation is probably most valuable as part of the ingest process. Governed by the archive's or collection's policy, submissions could be rejected or queued for administrative review if any Bitstreams do not pass validation for their identified formats. This helps ensure that the contents of those Bitstreams will be readable to users when disseminated, and that preservation operations (like migration) will be successful.

Note the similarity to monitoring quality-of-identification on ingest; a validation policy could be implemented by the same policy mechanism.

Migration, and Verifying Integrity of Format Migration

Archive administrators often rely on *migration* to preserve obsolete formats. It is also necessary to *verify* that a migration succeeded, i.e. compare the old and new versions of the Bitstream, making sure they are equivalent.

There are various techniques and software packages to migrate and verify digital objects. They can be modeled as application programs with an input `BitstreamFormat` (or range of formats), and an expected output `BitstreamFormat`.

The migration tool is matched to the subject of a migration by its input format, and by whether it can produce a non-obsolete output format.

The validation tool is matched to an existing pair of source and target Bitstreams, presumably the source and result of a migration. It must match both formats. It returns a Boolean value, true if the the target accurately represents the source. It may also generate a report or stream of warnings which should be handled the same way as *e.g.* warnings on ingest procedures.

In the DSpace configuration, migration and validation tools are listed with their input and output data formats described as namespaced external format identifiers. External format registries are the source of stable and persistent format identifiers.

Media Filter

The existing `MediaFilter` mechanism relies on data formats for two purposes: first, it looks for Bitstreams whose formats match the input format configured for each filter; and second, it depends on setting the format (as well as the name) of output Bitstreams to a certain known value which can be checked later to confirm that a filter has already been run.

Like the migration and verification tools, media filters are configured with formats named by stable and persistent format identifiers from external format registries.

The media filters in DSpace 1.4.x were configured and designed to work with fairly generic data formats, e.g. "PDF", but not a specific version of PDF. They should be mapped to similarly generic formats in the registries in use.

On the output side, the MediaFilter is configured with an external format identifier to impose on its output, and later to look for as a cue to detect the Bitstreams it created. (Though this is a poor technique for tracing its actions; administrative metadata documenting the relationship between source and output bitstreams would be easier to detect reliably and also more obvious to administrators and other applications.)