Importing and publishing assets via CSV

Importing Assets via CSV

This is a .csv template of an upload spreadsheet for assets. Here is documentation about the file path to use for asset uploads, including information about the transfer server and using Cyberduck. The process for bulk importing assets is to first import or create the items, then import the assets (which require parent items), then publish the assets (so that thumbnails are properly generated), then publish the items (requires a search of Hyacinth for the pids for these records in order to publish).

When you do a CSV asset import, you'll be using the following three columns in your CSV, with values for supplied for each CSV Asset row:

_import_file.import_type - the type of import, which we'll discuss in more detail later in this section
_import_file.import_path - the location of your file
_import_file.original_file_path - not required, but good to have if you want to maintain the original file structure of a directory, for a set of files. We'll also talk more about this later in the section.

Here's some info about the **import_type:**

- **external**
This is only available to Hyacinth administrators, and allows us to tell Hyacinth to create a new Asset record without actually moving the underlying file to a new location. Whatever you enter for _import_file.import_path will be used as the full path to the file. We call this "referencing an external file", thus the name "external." The reason this option isn't available to non-admins is because we don't want to allow people to track external files in any random location (like a local hard drive on a single server, or a network-mounted drive that isn't available on ALL servers). We want to guarantee that all files eventually wind up on one known, shared, high-integrity drive. At Columbia, this means our Isilon archival storage volume. We generally use **external** file imports for groups of files that have been run through Archivematica or have already been placed in their final archival storage location.

- **internal**
This import type is also only available to Hyacinth administrators. When used, the file at whatever full file path location you enter for _import_file.import_path will be COPIED to Hyacinth's internal storage, and the original file at _import_file.import_path will not be modified. This admin-only import type isn't used very often. It's kind of like a less restrictive, admin-only version of the "upload_directory" upload type that we'll discuss next.

- **upload_directory**
This is what non-admins use. This is a specific server directory (configured in Hyacinth's hyacinth.yml config file) that Hyacinth looks at for user uploads, and at Columbia we expect it to be available from most, if not all, of our servers (for upload convenience). It's currently located at "/ifs/cul/ldpd/hyacinth/uploads". Any files that you put in there (uploaded to a server via FTP) can be seen in the Hyacinth upload GUI or referenced in a CSV import using a **relative** path. So if you upload a file to "/ifs/cul/ldpd/hyacinth/uploads/myfolder/myfile.txt" and want to upload it via CSV, you'll have _import_file.import_type="upload_directory" and _import_file.import_path="myfolder/myfile.txt". So this **relative** path is the part that comes after "/ifs/cul/ldpd/hyacinth/uploads/" (or wherever the upload directory is). When you submit your CSV import job for processing, files in the upload directory will be *copied* into Hyacinth's internal storage. The original files will not be modified (just like if you had done an upload of type "internal"), so you should delete them from the upload directory after confirming that your batch has been successfully processed. In the future, we might automatically delete these files (as if they were **moved** to Hyacinth's internal storage rather than just **copied**), but for now that manual cleanup is necessary.

In the Hyacinth code base, there's technically a fourth upload type called "post_data", but that's really only used when doing a browser-based upload, so it doesn't ever apply to CSV imports.

Now let's talk a bit more about the **_import_file.original_file_path** CSV field. For ANY upload type, you may be interested in storing the original location of your file in a file hierarchy. If you don't supply an _import_file.original_file_path, Hyacinth will not be able to tell you the original location later on. Sometimes that data doesn't matter, but we've had projects where it does matter. It could be useful for generating a virtual filesystem view for a collection of objects later on. So let's say that you did an upload of a bunch of files, and your original hierarchy for those files looked like this:

/projectname/presentations/great-presentation.ppt
/projectname/presentations/another-presentation.ppt
/projectname/presentations/presentation-notes/all-presentation-notes.txt
/projectname/videos/video1.mp4
/projectname/videos/video2.mov

If you put these files in the upload directory and upload all of them, then each file will get stored in an internal storage layer that doesn't resemble the original system. By not including _import_file.original_file_path, you'd be choosing not to store the local hierarchy. But if you include the original file paths in your spreadsheet, then that information will be stored as metadata for future use and you'd be able to recreate that hierarchy of information.

A note for administrators who use the "external" file type: Just because you're supplying a full path to a file, that doesn't mean that we can guarantee that exact file path will be persisted forever. File systems and mount points change over time, and so do file storage layers (file storage vs. block storage, for example). This is why **the actual way we store files behind the scenes shouldn't matter to users**. The important thing is that we can reconstruct the file hierarchies from stored metadata later on, if necessary. So even when you're using type "external" or "internal" you still need to include a value for _import_file.original_file_path (unless you really really don't care about maintaining records about the original hierarchy of your files). Let's say that you're importing the same set of above files, but this time they're in their final "external" location on our archival storage volume, in the /digital/preservation/RBML directory, in an auto-named directory created by something like Archivematica :

/digital/preservation/RBML/bag-1234567/projectname/presentations/great-presentation.ppt
/digital/preservation/RBML/bag-1234567/projectname/presentations/another-presentation.ppt
/digital/preservation/RBML/bag-1234567/projectname/presentations/presentation-notes/all-presentation-notes.txt
/digital/preservation/RBML/bag-1234567/projectname/videos/video1.mp4
/digital/preservation/RBML/bag-1234567/projectname/videos/video2.mov

For your _import_file.original_file_path value, you do NOT want to include the "/digital/preservation/RBML/bag-1234567" part in your original file path. Imagine that these files came from a hard drive that was being archived. "/digital/preservation/RBML/bag-1234567" has nothing to do with that hard drive. You only want to use the part of the file paths that is relevant to the project itself, so your original file paths would look like:

/projectname/presentations/great-presentation.ppt
/projectname/presentations/another-presentation.ppt
/projectname/presentations/presentation-notes/all-presentation-notes.txt
/projectname/videos/video1.mp4
/projectname/videos/video2.mov

One other note about Asset creation: When importing files into Hyacinth to create Asset records, you can always import a new file, but you cannot update the file pointer for an existing Asset record to point to a different file. If you imported the wrong asset, you'd need to delete the old asset and create a new one.

Publishing via CSV

The general pattern for a publish target column heading is _publish_targets-X, with X being a number. So if you have a set of records that have up to three publish targets, you would have the following csv headers in your spreadsheet:

`_publish_targets-1.string_key`, `_publish_targets-2.string_key`, `_publish_targets-3.string_key`

The cell value for each of these columns would be a publish target string key (e.g. dlc_catalog, dlc_catalog_staging, sites, durst, carnegie).

When adding or removing publish targets, you also want to add another column heading to actually perform the publish action:

`_publish`

For every row that you want to publish, you'll put the value TRUE in this _publish column.

Although it's possible in the current version of Hyacinth (version 2) to add publish targets without also performing a publish action, this will change in Hyacinth 3. Hyacinth 3 will require the _publish column whenever you change the current publish targets for an object. In Hyacinth 2, adding publish targets without actually publishing has been a point of confusion on many occasions, since it makes it look like certain items *have* been published to targets even when they *haven't*.

Important note: If you do a CSV export of existing items from Hyacinth, and then edit that CSV and set new values for the `_publish_targets-X.string_key` fields, then you are overwriting the current set of publish targets. So if an object originally had three publish targets, but you submitted a CSV with only two _publish_targets-X.string key columns (e.g. `_publish_targets-1.string_key and _publish_targets-2.string_key),` you would be UN-publishing it from one of the three original publish targets.

Recognizing field blocks in CSV exports

Important note: If a field contains a colon (:), that means that the field is a part of a **field block**. If you update one of the fields in the block, you must include all the fields in the block in the import spreadsheet or Hyacinth will assume the other fields in the block have null value. This can cause you to lose data as empty columns overwrite existing data. For example, here is a field block of parent publication fields. You can tell they are in a block because they have the same prefix ("parent_publication-1:").

- parent_publication-1:parent_publication_doi
- parent_publication-1:parent_publication_issue
- parent_publication-1:parent_publication_page_end
- parent_publication-1:parent_publication_page_start
- parent_publication-1:parent_publication_title-1:parent_publication_title_sort_portion
- parent_publication-1:parent_publication_volume

If you would like to update information in one of these fields using a spreadsheet created by csv export, do NOT delete the other columns that are a part of your block. Leave the existing data in the other columns and include all the columns in your import spreadsheet.

See also: Updating Parent Objects (AKA _parent_digital_objects-1.pid)