CGIProposal

Issue Addressed

Configurable Submission is a very useful and popular feature of DSpace, but one quickly runs up against design limits. For example, the only mode of configurability for a given set of metadata inputs is by collection, whereas it might make more sense for the item's content type to determine it, or other factors.

General Problem

Configurable submission is one of only a few mechanisms in DSpace to assist the process of managing the metadata or other characteristics of items being ingested. Another tool is the Item template, which notably also is only configurable by collection. You can think of both of them in terms of a more general capability: using contextual information to streamline or automate the ingest. We can call this capability 'context-driven ingest' or perhaps 'context-assisted ingest'. Maybe the best description would be 'context-guided ingest' (CGI), since it suggests that context can provide relevant data, but need not rigidly determine all ingest properties.

Design Principles

What would a more flexible facility for CGI look like? We begin by examining a few general dimensions of the problem.

First, it is important to appreciate that the web submission UI is only one ingest pathway: context might also be relevant to administrative (batch) ingest, or non-interactive (e.g SWORD) deposits. Thus the first principle might be that the CGI solution not be tightly bound to submission code, but look more like a service that all ingest code can invoke.

Second, we should also observe that the means of specifying and defining the context must be kept distinct from the specification of the ingest properties themselves. An example can make this clear: suppose we want our context to suggest/force/default a particular value for a metadata field. So our 'ingest property' might be:

dc contributor author = jones

meaning that under the appropriate context, we default the author metadata to jones

Our specification should **not** include the context:

[monograph] dc contributor author = jones

because we may also want (under other circumstances):

[jones paper's collection] dc contributor author = jones

The point is simply that the mappings from context to ingest properties may be multiple and independent.

Third, we can consider how all these context mappings and ingest property sets are represented in the DSpace system. It is important to note that **none** of this data is real repository content (i.e. in the data model). We could delete it all, and no item in the repository would be affected. Thus, we should be wary of any design that requires a content API change of any sort.

Fourth, how will all this data (context mappings and ingest properties) be persisted? Most DSpace data lives in the asset store or RDBMS, but for configurable submission, it lives in an XML configuration file. As the number of different mappings grow, and more and more ingest properties are added, it will become less and less wieldy to manage in flat files, so a reasonable approach would be the database.