Migrating ManifoldCF installations across major versions

ManifoldCF maintains backwards compatibility, including automatic schema upgrades, across all software versions that share the same major version number. So, for example, if you are currently running MCF 1.7 and want to upgrade to MCF 1.10, all you need do is set up the new software and (depending on the model) either start it up, or run the "install" script. The same is true if you wanted to upgrade from MCF 2.0 to MCF 2.2.

However, automatic upgrades from one major version number to another major version number are not possible. The reason is that when such major version changes are made, backwards compatibility is explicitly broken. For example, MCF 2.0 removed significant functionality from many output and repository connectors, and instead placed that functionality into transformation connectors. This meant there was far less duplication of functionality, but it also meant that a simple upgrade is no longer possible. It is not even possible to export your 1.x configuration, and import it into 2.x. Instead, someone who wants to attempt this kind of upgrade must perform a manual migration.

This guide is meant to help people during the process of migrating their ManifoldCF applications from one major version to another. In addition to some overall general hints, there is also information about how specifically to perform the migration.

General hints

The first thing to plan out is what path your migration should take. When a new ManifoldCF major version comes along, the development team for a while creates pairs of releases – one for the old major version, and one for the new. It is thus possible to have (for instance) a 1.x release and a functionally comparable 2.x release working side by side. The first piece of advice, therefore, is to upgrade to the latest release that shares the same major version as the release you are currently using. After you confirm that that is working well, plan to migrate over to the functionally comparable equivalent MCF version with the newer major version; it will save you time and effort in the long run, and you may just be able to copy some things in the database as well. Finally, when that is working, plan to upgrade to the most recent release that shares the new major version.

The most recent pair of functionally comparable releases for 1.x and 2.x are MCF 1.10 and 2.2.

Of course, this path will also give you an opportunity to get any other supporting technology issues resolved. You may be using and older version of Tomcat, for instance, and you may want to know if that's something you need to upgrade. By going through the supported upgrade path on the older major version sequence, you can get to the point where it's possible to check this out without having to put a lot of effort into a migration that may not yet be possible.

Once you have planned your migration path and successfully upgraded to the desired older major version release, the next thing to figure out is whether you can save time on your migration by copying information from some of the older release database tables to the newer release database tables. Note that you will absolutely need to recreate all your connections and jobs on the new major release, regardless, so that would be the first step. See the sections below for hints about what may have changed that might require your specific attention. In general, you will want to bring up both the old major version's UI along with the new major version's UI. You should plan to go through the old major version's configuration tab by tab. In many cases it should be straightforward to copy the configuration, but you will also find that whole tabs are renamed, changed, or missing entirely. When that happens, it is a hint that you will need to do something else to create a comparable configuration on the new major version release. In general, you can ignore tabs that vanished that have no non-default configuration specified on them, but if that's not true, you will need to work a little harder to complete this task.

Once this is done, you may also be able to avoid recrawling everything if you are careful (and lucky). This is for advanced users, ONLY – those who are quite comfortable hacking the MCF backend database. (If you were using HSQLDB or Derby as your database choice, you are out of luck here – you won't be able to do anything more than recrawl using your new configuration). The goal is to save the data in the jobqueue and ingeststatus tables. Bear in mind that the data in these tables references job identifiers, which will have changed, so you will need not only to port the data over, but map the job identifiers from the old world to the new while you do it. Please also bear in mind that if you wound up changing your jobs in a significant way (by, for example, adding transformers to the pipeline to replace removed functionality), you are wasting your time trying to copy the data around; you'll wind up essentially recrawling everything anyway. But if your usage of MCF was very simple, there is a chance you can pull this off. In case you try to do this and fail, the worst you should need to do is to delete all rows in ingeststatus and jobqueue once again, and crawl from scratch.

Migrating from 1.x to 2.x

One of the major changes from 1.x to 2.x was the removal of the Forced Metadata tab that all jobs in 1.x had. This tab was replaced by a transformation connector, called the Metadata Adjuster. If any of your jobs used Forced Metadata, you will want to include the Metadata Adjuster in the job's pipeline.

This change also affected many output connectors that had metadata mapping functionality, such as Solr and ElasticSearch. This metadata mapping function was removed from the output connectors and moved to the Metadata Adjuster, so if you were using it, you will need to add the Metadata Adjuster in the pipeline. One instance of the Metadata Adjuster, though, should be enough to handle all cases.