Migration Data Work HOWTO / Toolkit

The following is for migrating into an existing system like PINES:

Get the incoming bib data, and translate to UTF-8 MARCXML. It may contain holdings. It may contain XML or MARC errors that you have to sanitize before your tools will work.

This is one way to translate MARC-8 MARC21 to UTF-8 MARCXML:
yaz-marcdump -f MARC-8 -t UTF-8 -o marcxml incoming.mrc > incoming.mrc.xml

> If you need to trim the bibs to a subset based on the presence of a certain value in a specific tag/subfield (for example, if you have the bibs for all libraries in a foreign system and only need bibs belonging to a specific migrating library, you might filter based on their holding tags)
> **trim_marc_based_on_tag_subfield_value.pl 999 m BRANCH_CODE incoming.mrc.xml > incoming.filtered.mrc.xml**

Embed potential native record ids into the incumbent records

set_record_ids.pl 100000 903 a incoming.mrc.xml > incoming.renumbered.mrc.xml

Get primary fingerprints for incoming data and get a bib dump of matching records from the incumbent system

fingerprints.pl primary 903 a incoming.renumbered.mrc.xml > incoming.primary.fp 2> incoming.primary.fp_err

Edit the query_for_primary_matching_incumbent_record.pl script to point to the correct Evergreen database and table holding the incumbent primary fingerprints (FIXME add in how to create such a table).
query_for_primary_matching_incumbent_record.pl incoming.primary.fp | sort | uniq > primary_matching_incumbent.record_ids

In a postgres shell, you create a temporary table to hold these id's:
CREATE TABLE primary_matching_incumbent_records_for_incoming_library (id BIGINT);
COPY primary_matching_incumbent_records_for_incoming_library FROM 'primary_matching_incumbent.record_ids';

To dump the matching incumbent records to a file, in a postgres shell do:

\t
\o matching_incumbent_records.dump
select b.id, b.tcn_source, b.tcn_value, regexp_replace(b.marc,E'\n','','g') from biblio.record_entry as b join primary_matching_incumbent_records_for_incoming_library as c using (id) ;

Now to turn that dump into a MARCXML file with record numbers and TCN embedded in tag 901, do:
marc_add_ids -f id -f tcn_source -f tcn_value -f marc < matching_incumbent_records.dump > matching_incumbent_records.mrc.xml

> It's possible that this file may need to be itself sanitized some. This will transform code="""" into code="&x0022;", for example:
> **cat matching_incumbent_records.mrc.xml | sed 's/code=\"\"\"/ code=\"\"\"/' > matching_incumbent_records.escaped.mrc.xml**

Get full fingerprints for both datasets and match them.

fingerprints.pl full 901 c matching_incumbent_records.mrc.xml > incumbent.fp 2> incumbent.fp_err
fingerprints.pl full 903 a incoming.renumbered.mrc.xml > incoming.fp 2> incoming.fp_err

The script below will produce matched groupings, and can optionally take a 4th and 5th parameter providing scoring information for determining lead records. In the past, this would consider certain metrics for MARC quality, but in the latest incarnation, it assumes an incumbent record will be the lead record, and looks at # of holdings and possible matching of tag 245 subfield b for determining which of the incumbent records would be the lead record. The example invocation below does not use scoring.

match_fingerprints.pl "name of dataset for dedup interface" incumbent.fp incoming.fp

This will produce two files, match.groupings and match.record_ids. The format for match.groupings is suitable for insertion into the db for the dedup interface.

Import these matches and records into the legacy dedup interface for viewing:

Now to tar up the specific MARC records involved for the dedup interface:

cat match.groupings | cut -d^ -f3 > incumbent.record_ids
cat match.groupings | cut -d^ -f5 | cut -d, -f2- | sed 's/,/\n/g' > incoming.record_ids

mkdir dataset ; cd dataset
select_marc.pl ../incumbent.record_ids 901 c ../matching_incumbent_records.mrc.xml
select_marc.pl ../incoming.record_ids 903 a ../incoming.renumbered.mrc.xml
cd ..
tar cvf dataset.tar dataset

In a mysql shell for the database used with the dedup interface:
LOAD DATA LOCAL INFILE 'match.groupings' INTO TABLE record_group FIELDS TERMINATED BY '^' (status, dataset, best_record,records,original_records);

Create a pretty printed text dump of the non-matching incoming records:

dump_inverse_select_marc.pl incoming.record_ids 903 a incoming.renumbered.mrc.xml > non_matching_incoming.mrc.txt 2> non_matching_incoming.mrc.txt.err