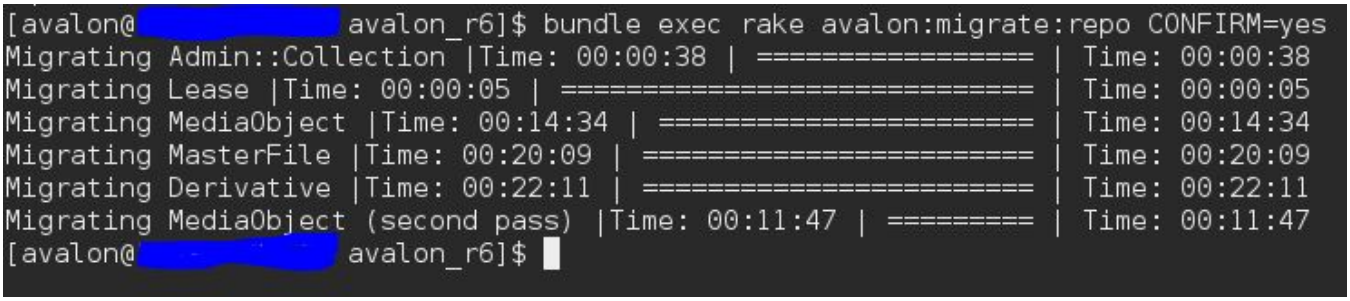
Migration Tool

The Avalon 5.x to 6.x migration tool copies and transforms Avalon 5.x's Fedora 3 data to Avalon 6.x's Fedora 4 data. The tool can be run as follows:

```
bundle exec rake avalon:migrate:repo
```

When you run the migration in the command line, you can see a broad view of how much progress has been made for each Class. Notice the last line indicates a second pass for *Migrating MediaObject*. The second pass is necessary to associate master files with media objects and get them ordered properly.



Command Options

Option	Explanation	Example Usage	Default Value
CONFIRM	Required in order to prevent accidental execution	CONFIRM=yes	
pids	A comma separated list of Avalon 5.x Fedora 3 pids to migrate	pids=avalon:1, avalon:2,avalon:3	
pidfile	Path to a filename containing one Avalon 5.x Fedora 3 pid per line	pidfile=/path/to /pidfile	
parallel_processes	Number of parallel processes the migration tool should use	parallel_processes=4	number of processors - 2
overwrite	Determines whether: existing Fedora 4 objects should be kept, cleaned out, and reused (overwrite=true); or new objects should be created even if matching migrated objects already exist (overwrite=false)	overwrite=true	false
namespace	The Fedora 3 namespace that should be migrated	namespace=media	avalon
skip_completed	Skip items with status = completed	skip_completed=true	false

Rerun a migration to bring in new data

You'll want to set skip_completed and overwrite to true. Refer to the logic here

```
bundle exec rake avalon:migrate:repo overwrite=true skip_completed=true CONFIRM=yes
```

```
    def migration_required?(pid, klass, method=:migrate)
      status_report = MigrationStatus.find_by(source_class: klass, f3_pid: pid, datastream: nil)
      status_report.nil? ||
        (status_report.status != 'completed' && status_report.status != 'waiting' && method == :migrate) ||
        (status_report.status != 'completed' && method == :second_pass) ||
        (!skip_completed? && overwrite?)
    end
```

Rerunning the migration to fix failed objects

Once the migration is running, it is best to let it fully finish before attempting to rerun failed objects. The migration should fail objects that are missing parent or child objects. Once the migration is finished running, a list of the failed object ids can be collected into a file by running this command:

```
bundle exec rails r 'puts MigrationStatus.where(status: "failed").map(&:f3_pid).uniq.join("\n")' >
failed_objects
```

The migration tool can then be rerun with this list:

```
bundle exec rake avalon:migrate:repo pidfile=failed_objects
```

Potential Problems

First read the Migration Report to determine what objects failed migration and why.

Problem	Workaround
Objects fail with errors about mismatched prefixes	Turn off parallel processing by setting parallel_processes=1
Objects fail with LDP conflict	Turn off parallel processing by setting parallel_processes=1
Poster and thumbnail images missing	See script on Miscellaneous scripts for R5/R6 migration
Section permalinks incorrect	See script on Miscellaneous scripts for R5/R6 migration
Permalink collisions	See script on Miscellaneous scripts for R5/R6 migration

Helper Scripts

See Miscellaneous scripts for R5/R6 migration for scripts that can be adapted to your institution to help find and fix data issues post-migration.