Miscellaneous scripts for R5/R6 migration

This page contains miscellaneous scripts that were used for data verification and correction during the process and in the aftermath of upgrading from R5 to R6. This scripts are for reference only and not applicable generally, but rather they serve as examples of how to work with the data in R5 and R6.

Missing Posters

After migration we noticed that some of our videos didn't have thumbnails or posters. We ran these scripts to check the status of these items in R6. We noticed they either didn't have derivatives, or they had a duration of 0. So we checked the status of these items in R5 and found that they had the same issues there.

```
Scripts I ran to determine state of problem masterfiles:

# On Fedora 4 system, find migrated MasterFiles that don't have posters or durations

mfs = []
durations = []
count = 0
MasterFile.find_each({},{batch_size:5}) do |mf|
  count += 1
  mfs << mf.id unless mf.file_format == 'Sound' or mf.has_poster?
  durations << mf.id if mf.duration.nil?
  if count%200 == 0
    puts " #{count} - #{mfs.count}, #{durations.count}"
  else
    print "."
  end
end

# Get the Fedora 3 pids for those items

s = MigrationStatus.where( f4_pid: mfs|durations )
s.each do |s|
  f3s << s.f3_pid
end

f3s

# On Fedora 3 system, see if those MasterFiles had derivatives/duration before migrating

f3s.each do |id|
  mf = MasterFile.find(id)
  puts "#{id}: #{mf.duration} #{mf.derivatives.count}"
end
```

Permalink Collisions

For some reason, after migrating we had permalinks that pointed to more than one MasterFile. We ran this script to identify those collisions and then fix them.

```
#!/usr/bin/env ruby
require 'open-uri'
require 'nokogiri'
def report(collisions)
  puts "Found #{collisions.keys.size} Collisions" unless collisions.empty?
  collisions.each do |id,field_values|
    puts "-------------"
    puts "#{id} ->"
    field_values.each {|doc| puts doc}
  end
  puts "-------------"
  f4_ids = collisions.values.collect {|arr| arr.collect {|h| h[:id]}}.flatten.uniq
  puts "Collisions involve #{f4_ids.size} objects:"
  puts f4_ids.inspect
end
result = Nokogiri::XML(open("http://localhost:8983/solr/avalon/select?q=*&rows=0&facet=on&facet.
field=identifier_ssim&facet.limit=-1&facet.mincount=2"))
```

```
collided_ids = result.xpath('//lst[@name="identifier_ssim"]/int/@name').collect(&:value)
collisions = {}
collided_ids.each do |id|
  fields = ["id", "has_model_ssim", "system_create_dtsi", "system_modified_dtsi"]
  collided_docs = Nokogiri::XML(open("http://localhost:8983/solr/avalon/select?q=identifier_ssim:#{id}&fl=#
{fields.join(',')}"))
  collided_docs.xpath('//doc').each do |doc|
    field_values = {}
    fields.each {|f| field_values[f.to_sym] = doc.xpath("*[@name='#{f}']").text }
    obj_id = field_values[:id]
    collisions[id] ||= []
    collisions[id] << field_values
  end
end
report(collisions)

# use the above to generate a list of collisions, then munge the into this form:
# h={permalink_id1: [masterfile_id1, masterfile_id2], ... }
# once your h hash looks good, pass it to split_mfs to correct collisions
def split_mfs h
  mo_cache = {}
  ids_cache = {}
  h.values.each do |vals|
    mf1, mf2 = vals
    m1 = MasterFile.find(mf1) rescue nil
    m2 = MasterFile.find(mf2) rescue nil
    good_mf = nil
    bad_mf = nil
    mo = nil
    print "#{m1.id} (1) / #{m2.id} (2): MediaObject "

    if m1.derivatives.count > 0
      good_mf = m1
      bad_mf = m2
    elsif m2.derivatives.count > 0
      good_mf = m2
      bad_mf = m1
    end

    if good_mf.present? and bad_mf.present?

      mo_id = good_mf.media_object_id || bad_mf.media_object.id
      mo_cache[mo_id] ||= MediaObject.find(mo_id) rescue nil
      mo = mo_cache[mo_id]

      if mo.present?
        print "#{mo_id} "
        ids_cache[mo_id] ||= mo.master_files.collect(&:id)
        if ids_cache[mo_id].include? good_mf.id
          puts " correctly associated with good mf #{good_mf.id}, deleting bad_mf #{bad_mf.id}"
          bad_mf.delete
        elsif ids_cache[mo_id].include? bad_mf.id
          mf_index = mo.ordered_master_file_ids.index bad_mf.id
          puts " dropping association with and deleting bad_mf #{bad_mf.id} at index #{mf_index}. Associating
good_mf #{good_mf.id}"
          good_mf.media_object_id = mo_id
          good_mf.save!
          mo.ordered_master_files.delete_at( mf_index )
          mo.ordered_master_files.insert_at( mf_index, good_mf )
          mo.master_files -= [bad_mf]
          mo.save!
          bad_mf.delete
        else
          puts " not associated with either mf #{good_mf.id} or #{bad_mf.id}"
        end
      else
        puts " media_object not found"
      end
    else
      puts " derivatives not found "
    end
```

```
  end
end
```

Update Permalinks

update_permalinks.rb

```
def permalink_https!(obj)
  permalink_uri = URI.parse(obj.permalink)
  if permalink_uri.scheme == 'http'
    permalink_uri.scheme = 'https'
    obj.permalink = permalink_uri.to_s
    obj.save!
  else
    false
  end
end

#puts "Moving media object permalinks to https"
#count = 0
#MediaObject.find_each({},{batch_size: 5}) do |obj|
#  begin
#    result = permalink_https!(obj)
#    if result
#      count = count + 1
#      puts "#{count}: Updated #{obj.id}: #{obj.permalink}"
#    else
#      puts "#{count}: Skipping #{obj.id}: #{obj.permalink}"
#    end
#  rescue URI::InvalidURIError => e
#     puts "Failed for #{obj.id}: #{obj.permalink}"
#     puts e.backtrace
#  end
#end
#puts "Solr commit and optimize"
#ActiveFedora::SolrService.instance.conn.commit
#ActiveFedora::SolrService.instance.conn.optimize
puts "Moving master file permalinks to https"
count = 0
MasterFile.find_each({},{batch_size:5}) do |obj|
  begin
    result = permalink_https!(obj)
    if result
      count = count + 1
      puts "#{count}: Updated #{obj.id}: #{obj.permalink}"
    else
      puts "#{count}: Skipping #{obj.id}: #{obj.permalink}"
    end
  rescue URI::InvalidURIError => e
     puts "Failed for #{obj.id}: #{obj.permalink}"
     puts e.backtrace
  end
end
puts "Solr commit and optimize"
ActiveFedora::SolrService.instance.conn.commit
ActiveFedora::SolrService.instance.conn.optimize
puts "Done."
```

Reindex PURLS

reindex_for_case_insensitive_purls.rb

```
puts "Reindexing media objects"
count = 0
MediaObject.find_each({},{batch_size: 5}) do |obj|
  obj.update_index
  count = count + 1
  puts "#{count}: Updated index for #{obj.id}: #{obj.identifier.join(', ')}"
end
puts "Solr commit and optimize"
ActiveFedora::SolrService.instance.conn.commit
ActiveFedora::SolrService.instance.conn.optimize
puts "Reindexing master files"
count = 0
MasterFile.find_each({},{batch_size: 5}) do |obj|
  obj.update_index
  count = count + 1
  puts "#{count}: Updated index for #{obj.id}: #{obj.identifier.join(', ')}"
end
puts "Solr commit and optimize"
ActiveFedora::SolrService.instance.conn.commit
ActiveFedora::SolrService.instance.conn.optimize
puts "done"
```

Permalink Validation

Script to double check that permalinks for MediaObjects and MasterFiles ended up with the correct values and in the correct places after migration.

```
load '/srv/avalon/avalon_r6/f3_mopermalinks.rb' # includes F3P = { f3_pid: permalink }
load '/srv/avalon/avalon_r6/f3_mfpermalinks.rb' # includes F3MFP = { f3_pid: permalink }

puts 'Inspecting F4 permalinks for MediaObjects'
puts '1) If permalink in Fedora 3, the Fedora 4 object should have the same permalink'
puts '2) If permalink in Fedora 4, the permalink noid should be persisted in Fedora 4 in identifiers:local '
puts '3) If permalink in Fedora 4, the permalink noid should be persisted in Solr in identifer_ssim '

count = 0

MediaObject.find_each({},{batch_size: 5}) do |obj|
  count += 1
  migration_status = MigrationStatus.where(f4_pid: obj.id).first
  unless migration_status.nil?
    f3_permalink = F3P[migration_status.f3_pid]
    unless f3_permalink.nil?
      unless f3_permalink.split('://').last == obj.permalink.split('://').last
        puts "#{obj.id} F3/F4 permalink mismatch: #{f3_permalink} / #{obj.permalink}"
      end
    end
  end
  if obj.permalink.present?
    unless obj.identifier.include? obj.permalink.split('/').last
      puts "#{obj.id} F4 permalink not found in identifiers"
    end
    unless obj.to_solr['permalink_tesim'].include? obj.permalink
      puts "#{obj.id} F4 permalink not found in solr"
    end
  end
  print '.' if (count%10==0)
  puts count if (count%1000==0)
  STDOUT.flush if (count%10==0)
end

count = 0

puts ''
puts 'Inspecting F4 permalinks for MasterFiles'

MasterFile.find_each({},{batch_size: 5}) do |obj|
  count += 1
  migration_status = MigrationStatus.where(f4_pid: obj.id).first
  unless migration_status.nil?
    f3_permalink = F3MFP[migration_status.f3_pid]
    unless f3_permalink.nil?
      unless f3_permalink.split('://').last == obj.permalink.split('://').last
        puts "#{obj.id} F3/F4 permalink mismatch: #{f3_permalink} / #{obj.permalink}"
      end
    end
  end
  if obj.permalink.present?
    unless obj.identifier.include? obj.permalink.split('/').last
      puts "#{obj.id} F4 permalink not found in identifiers"
    end
    unless obj.to_solr['permalink_tesim'].include? obj.permalink
      puts "#{obj.id} F4 permalink not found in solr"
    end
  end
  print '.' if (count%10==0)
  puts count if (count%1000==0)
  STDOUT.flush if (count%10==0)
end
```