Batch Ingesting into EPrints Digital Repository Software

Tomasz Neugebauer and Bin Han

ABSTRACT

This paper describes the batch importing strategy and workflow used for the import of theses metadata and PDF documents into the EPrints digital repository software. A two-step strategy of importing metadata in MARC format followed by attachment of PDF documents is described in detail, including Perl source code for scripts used. The processes described were used in the ingestion of 6,000 theses metadata and PDFs into an EPrints institutional repository.

INTRODUCTION

Tutorials have been published about batch ingestion of ProQuest metadata and electronic theses and dissertations (ETDs),[1] as well as EndNote library,[2] into the Digital Commons platform. The procedures for bulk importing of ETDs using DSpace have also been reported.[3] However, bulk importing into the EPrints digital repository software has not been exhaustively addressed in the literature.[4] A recent article by Walsh provides a literature review of batch importing into institutional repositories.[5] The only published report on batch importing into the EPrints platform describes Perl scripts for metadata-only records import from Thomson Reuters Reference Manager.[6]

Bulk importing is often one of the first tasks after launching a repository, so it is unsurprising that requests for reports and documentation on EPrints-specific workflow have been a recurring question on the EPrints Tech List.[7] A recently published review of EPrints identifies "the absence of a bulk uploading feature" as its most significant weakness.[8] Although EPrints' graphical user interface for bulk importing is limited to the use of the installed import plugins, the software does have a versatile infrastructure for this purpose. Leveraging EPrints' import functionality requires some Perl scripting, structuring the data for import, and using the command line interface.

In 2009, when Concordia University launched Spectrum,[9] its research repository, the first task was a batch ingest of approximately 6,000 theses dated from 1967 to 2003. The source of the metadata for this import consisted in MARC records from an integrated library system powered by Innovative Interfaces and ProQuest PDF documents. This paper is a report on the strategy and workflow adopted for batch ingestion of this content into the EPrints digital repository software.

Import Strategy

EPrints has a documented import command line utility located in the */bin* folder.[10] Documents can also be imported through EPrints' graphical interface. Using the command line utility for

Tomasz Neugebauer *(tomasz.neugebauer@concordia.ca) is Digital Projects and Systems Development Librarian and* ***Bin Han*** *(bin.han@concordia.ca) is Digital Repository Developer, Concordia University Libraries, Montreal, Quebec, Canada.*

importing is recommended because it is easier to monitor the operation in real time by adding progress information output to the import plugin code.

The task of batch importing can be split into the following subtasks:

1. import of metadata of each item
2. import of associated documents, such as full-text PDF files

The strategy adopted was to first import the metadata for all of the new items into the inbox of an editor's account. After this first step was completed, a script was used to loop through the newly imported eprints and attach the corresponding full-text documents. Although documents can be imported from the local file system or via HTTP, import of the files from the local file system was used.

The batch import procedure varies depending on the format of the metadata and documents to be imported. Metadata import requires a mapping of the source schema fields to the default or custom fields in EPrints. The source metadata must also be converted into one of the formats supported by EPrints' import plugins, or a custom plugin must be created. Import plugins are available for many popular formats, including BibTeX, DOI, EndNote, and PubMedXML. In addition, community-contributed import plugins such as MARC and ArXiv are available at EPrints Files.[11] Since most repositories use custom metadata fields, some customization of the import plugins is usually necessary.

MARC Plugin for EPrints

In EPrints, the import and export plugins ensure interoperability of the repository with other systems. Import plugins read metadata from one schema and load it into the EPrints system through a mapping of the fields into the EPrints schema. Loading MARC-encoded files into EPrints requires the installation of the import/export plugin developed by Romero and Miguel.[12] The installation of this plugin requires the following two CPAN modules: *MARC::Record* and *MARC::File::USMARC*. The MARC plugin was then subclassed to create an import plugin named "Concordia Theses," which is customized for thesis MARC records.

Concordia Theses MARC Plugin

The MARC plugin features a central configuration file (see appendix A) in which each MARC field is paired with a corresponding mapping to an EPrints field. Most of the fields were configured through this configuration file (see table 1).

The source MARC records from the Innovative Interfaces Integrated Library System (ILS) encode the physical description of each item using the Anglo American Cataloguing Rules, as in the following example: "ix, 133 leaves : ill. ; 29 cm." Since the default EPrints field for number of pages is of the type integer and does not allow multipart physical descriptions from the MARC 300 field, a custom text field for these physical descriptions (pages_aacr) had to be added.

The *marc.pl* configuration file cannot be used to map compound fields, such as author names—the fields need custom mapping implementation in Perl. For instance, the MARC 100 and 700 fields

are transferred into the EPrints author compound field (in *MARC.pm*). Similarly, MARC 599 is mapped into a custom thesis advisor compound field.

MARC field	EPrints field
020a	isbn
020z	isbn
022a	issn
245a	title
250a	edition
260a	place_of_pub
260b	publisher
260c	date
300a	pages_aacr
362a	volume
440a	series
440c	volume
440x	issn
520a	abstract
730a	publication

Table 1. Mapping Table from MARC to EPrints

Helge Knüttel's refinements to the MARC plugin shared on the EPrints Tech List were employed in the implementation of a new subclass of MARC import for the Concordia Theses MARC records. In the implementation of the Concordia Theses plugin, *ConcordiaTheses.pm* inherits from *MARC.pm.* (See figure 1.)[13]

Knüttel added two methods that make it easier to subclass the general MARC plugin and add unique mappings: *handle_marc_specialities* and *post_process_eprint*. The *post_process_eprint* function was not used to attach the full-text documents to each eprint. Instead, the strategy to import the full-text documents using a separate *attach_documents* script was used (see "Theses Document File Attachment" below). Import of all of the specialized fields, such as *thesis type* (mapped from MARC 710t), *program*, *department*, and *proquest id*, was implemented in the function *handle_marc_specialities* of *ConcordiaTheses.pm*. For instance, 502a in the MARC record contains the department information, whereas an EPrints system like Spectrum stores department hierarchy as subject objects in a tree. Therefore importing the department information based on the value of 502a required regular expression searches of this MARC field to find the mapping into a corresponding *subject id.* This was implemented in the *handle_marc_specialities* function.

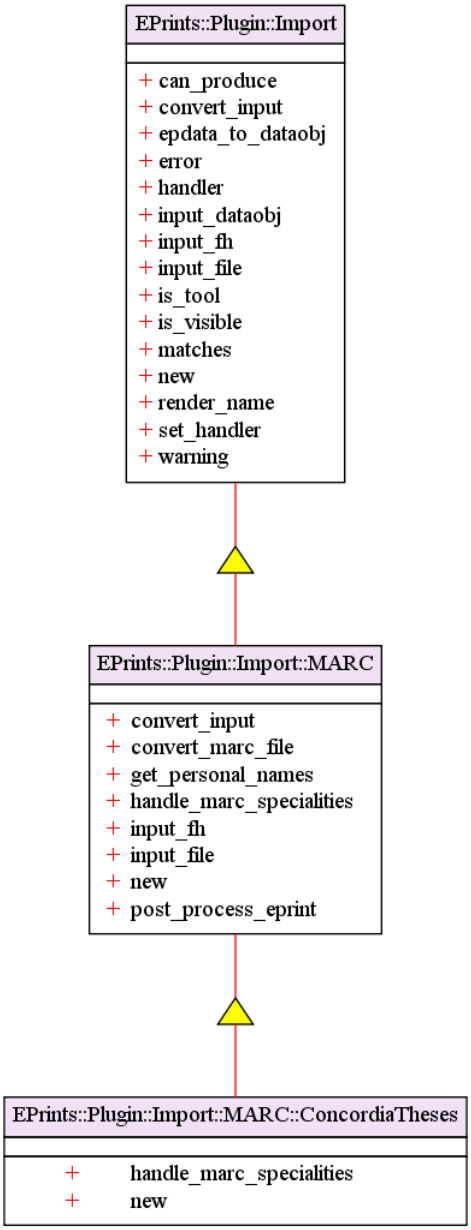

Figure 1. Concordia Theses Class Diagram, created with the Perl module UML::Class::Simple

Execution of the Theses Metadata Import

The depositing user's name is displayed along with the metadata for each eprint. A *batchimporter* user with the corporate name "Concordia University Libraries" was created to carry out the import. As a result, the public display of the imported items shows the following as a part of the metadata: "Deposited By: Concordia University Libraries." The MARC plugin requires the encoding of the source MARC files to be UTF-8, whereas the records are exported from the ILS with MARC-8 encoding. Therefore MarcEdit software developed by Reese was used to convert the MARC file to UTF-8.[14]

To activate the import, the main MARC import plugin and its subclass, *ConcordiaTheses.pm*, have to be placed in the plugin folder */perl_lib/EPrints/Plugin/Import/MARC/*. The configuration file

(see appendix A) must also be placed with the rest of the configurable files in */archives/REPOSITORYID/cfg/cfg.d*. The plugin can then be activated from the command line using the *import* script in the */bin* folder. A detailed description of this script and its usage is documented on the EPrints Wiki. The following EPrints command from the */bin* folder was used to launch the import:

> *import REPOSITORYID --verbose --user batchimporter eprint MARC::ConcordiaTheses Theses-utf8.mrc*

Following the aforementioned steps, all the theses metadata was imported into the EPrints software. The new items were imported with their statuses set to *inbox*. A status set to *inbox* means that the imported items are in the work area of *batchimporter* user and will need to be moved to live public access by switching their status to *archive*.

Theses Document File Attachment

After the process of importing the metadata of each thesis is complete, the corresponding document files need to be attached. The *proquest id* was used to link the full-text PDF documents to the metadata records. All of the MARC records contained the *proquest id*, while the PDF files, received from ProQuest, were delivered with the corresponding *proquest id* as the filename. The PDFs were uploaded to a folder on the repository web server using FTP. The *attach_documents* script (see appendix B for source code) was then used to attach the documents to each of the imported eprints in the *batchimporter*'s inbox and to move the imported eprints to the live archive.

Several variables need to be set at the beginning of the *attach_documents* operation (see table 2).

Variable	Comment
$root_dir = 'bin/import-data/proquest'	This is the root folder where all the associated documents are uploaded by FTP.
$depositor = 'batchimporter'	Only the items deposited by a defined depositor, in this case batchimporter, will be moved from inbox to live archive.
$dataset_id = 'inbox'	Limit the dataset to those eprints with status set to inbox
$repositoryid = 'library'	The internal EPrints identifier of the repository

Table 2. Variables to be Set in the attach_documents Script

The following command is used to proceed with file attachment, while the output log is redirected and saved in the file *ATTACHMENT:*

/bin/attach_documents.pl > ./ATTACHMENT 2>&1

The thesis metadata record was made live even if it did not contain a corresponding document file. A list of *eprint ids* of theses that did not contain a corresponding full-text PDF document are listed at the end of the log file, along with the count of the number of theses that were made live.

After the import operation is complete, all the abstract pages need to be regenerated with the following command:

/bin/generate_abstracts REPOSITORYID

CONCLUSIONS

This paper is a detailed report on batch importing into the EPrints system. The authors believe that this paper and its accompanying source code is a useful contribution to the literature on batch importing into digital repository systems. In particular, it should be useful to institutions that are adopting the EPrints digital repository software. Batch importing of content is a basic and fundamental function of a repository system, which is why the topic has come up repeatedly on the EPrints Tech List and in a repository software review.

The methods that we describe for carrying out batch importing in EPrints make use of the command line and require Perl scripting. More robust administrative graphical user interface support for batch import functions would be a useful feature to develop in the platform.

ACKNOWLEDGEMENTS

The authors would like thank Mia Massicotte for exporting the metadata records from the integrated library system. We would also like to thank Alexandros Nitsiou, Raquel Horlick, Adam Field, and the reviewers at Information Technology and Libraries *for their useful comments and suggestions.*

REFERENCES

1. Shawn Averkamp and Joanna Lee, "Repurposing ProQuest Metadata for Batch Ingesting ETDs into an Institutional Repository," *code{4}lib journal* 7 (2009), http://journal.code4lib.org/articles/1647 (accessed June 27, 2011).
2. Michael Witt and Mark P. Newton, "Preparing Batch Deposits for Digital Commons Repositories," 2008, http://docs.lib.purdue.edu/lib_research/96/ (accessed June 20, 2011).
3. Randall Floyd, "Automated Electronic Thesis and Dissertations Ingest," 2009, https://wiki.dlib.indiana.edu/display/IUSW/Automated+Electronic+Thesis+and+Dissertations+Ingest (accessed May 26, 2011).
4. EPrints Digital Repository Software, University of Southampton, UK, http://www.eprints.org/ (accessed June 27, 2011).
5. Maureen P. Walsh, "Batch Loading Collections into DSpace: Using Perl Scripts for Automation and Quality Control," *Information Technology & Libraries* 29, no. 3 (2010): 117–27,

http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=52871761&site=ehost-live (accessed June 26, 2011).
6. Lesley Drysdale, "Importing Records from Reference Manager into GNU EPrints," 2004, http://hdl.handle.net/1905/175 (accessed June 27, 2011).
7. EPrints Tech List, University of Southampton, UK, http://www.eprints.org/tech.php/ (accessed June 27, 2011).
8. Mike Beazly, "Eprints Institutional Repository Software: A Review," *Partnership: the Canadian Journal of Library & Information Practice & Research* 5, no. 2 (2010), http://journal.lib.uoguelph.ca/index.php/perj/article/viewArticle/1234 (accessed June 27, 2011).
9. Concordia University Libraries, "Spectrum: Concordia University Research Repository," http://spectrum.library.concordia.ca (accessed June 27, 2011).
10. EPrints Wiki, "API:bin/import," University of Southampton, UK, http://wiki.eprints.org/w/API:bin/import (accessed June 23, 2011).
11. EPrints Files, University of Southampton, UK, http://files.eprints.org/ (accessed June 24 2011).
12. Parella Romero and Jose Miguel, "MARC Import/Export Plugins for GNU EPrints3," EPrints Files, 2008, http://files.eprints.org/323/ (accessed May 31, 2011).
13. Agent Zhang and Maxim Zenin, "UML:Class::Simple," CPAN, http://search.cpan.org/~agent/UML-Class-Simple-0.18/lib/UML/Class/Simple.pm (accessed September 20, 2011).
14. Terry Reese, "MarcEdit: Downloads," Oregon State University, http://people.oregonstate.edu/~reeset/marcedit/html/downloads.html (accessed June 27, 2011).

Appendix A. marc.pl Configuration File

```
#
# Plugin EPrints::Plugin::Import::MARC
#
# MARC tofro EPrints Mappings
# Do _not_ add compound mappings here.
$c->{marc}->{marc2ep} = { # MARC to EPrints
   '020a' => 'isbn',
   '020z' => 'isbn',
   '022a' => 'issn',
   '245a' => 'title',
   '245b' => 'subtitle',
   '250a' => 'edition',
   '260a' => 'place_of_pub',
   '260b' => 'publisher',
   '260c' => 'date',
   '362a' => 'volume',
   '440a' => 'series',
   '440c' => 'volume',
   '440x' => 'issn',
   '520a' => 'abstract',
   '730a' => 'publication',
};

$c->{marc}->{marc2ep}->{constants} = {
};

######################################################################
###
#
# Plugin-specific settings.
#
# Any non empty hash set for a specific plugin will override the
# general one above!
#
######################################################################
###
#
# Plugin EPrints::Plugin::Import::MARC::ConcordiaTheses
#
$c->{marc}->{'EPrints::Plugin::Import::MARC::ConcordiaTheses'}->{marc2ep} = {
   '020a' => 'isbn',
   '020z' => 'isbn',
   '022a' => 'issn',
   '250a' => 'edition',
```

```
  '260a' => 'place_of_pub',
  '260b' => 'publisher',
  '260c' => 'date',
  '300a' => 'pages_aacr',
  '362a' => 'volume',
  '440a' => 'series',
  '440c' => 'volume',
  '440x' => 'issn',
  '520a' => 'abstract',
  '730a' => 'publication',
};

$c->{marc}->{'EPrints::Plugin::Import::MARC::ConcordiaTheses'}->{constants} = { # MARC to
EPrints constants
  'type' => 'thesis',
  'institution' => 'Concordia University',
  'date_type' => 'submitted',
};
```

Appendix B. attach_documents.pl

```
#!/usr/bin/perl -I/opt/eprints3/perl_lib

=head1 DESCRIPTION

This script allows you to attach a file to an eprint object by proquest id.

=head1 COPYRIGHT AND LICENSE
 2009 Adam Field, Tomasz Neugebauer <tomasz.neugebauer@concordia.ca>
 2011 Bin Han <bin.han@concordia.ca>
This module is free software under the same terms of Perl.
Compatible with EPrints 3.2.4 (Victoria Sponge).

=cut

use strict;
use warnings;
use EPrints;

my $repositoryid = 'library';
my $root_dir = '/opt/eprints3/bin/import-data/proquest'; #location of PDF files
my $dataset_id = 'inbox';   #change to 'eprint' if you want to run it over everything.
my $depositor = 'batchimporter';    #limit import to $depositor's Inbox

#global variables for log purposes
my $int_live = 0;      #count of eprints moved to live archive with a document
my $int_doc = 0;       #count of eprints that already have document attached
my @array_doc;         #ids of eprints that already have documents
my $int_no_doc = 0;    #count of eprints moved to live with no document attached
my @array_no_doc;      #ids of eprints that have no documents
my $int_no_proid = 0;  #count of eprints with no proquest id
my @array_no_proid;    #ids of eprints with no proquest id

my $session = EPrints::Session->new(1, $repositoryid);
die "couldn't create session for $repositoryid\n" unless defined $session;

#the hash contains all the files that need to be uploaded
#the hash contains key-value pairs: (pq_id => filename)
my $filemap = {};
load_filemap($root_dir);

#get all eprints in inbox dataset
my $dataset = $session->get_repository->get_dataset($dataset_id);
#run attach_file on each eprint object
$dataset->map($session, \&attach_file);
```

```
#output log for attachment
print "#### $int_doc eprints already have document attached, skip ####\n @array_doc\n";
print "#### $int_no_proid eprints doesn't have proquest id, skip ####\n @array_no_proid\n";
print "#### $int_no_doc eprints doesn't have associated document, moved to live ####\n
@array_no_doc\n";

#total number of eprints that were made live: those with and without documents.
my $int_total_live = $int_live + $int_no_doc;
print "#### Intotal: $int_total_live eprints moved to live ####\n";

#attach file to corresponding eprint object
sub attach_file
{
  my ($session, $ds, $eprint) = @_;

  #skip if eprint already has a document attached
  my $full_text_status = $eprint->get_value( "full_text_status" );
  if ($full_text_status ne "none")
  {
    print "EPrint ".$eprint->get_id." already has a document, skipping\n";
    $int_doc ++;
    push ( @array_doc, $eprint->get_id );
    return;
  }


  #retrieve username/userid associated with current eprint
  my $user = new EPrints::DataObj::User(
    $eprint->{ session },
    $eprint->get_value( "userid" ) );
  my $username;

  # exit in case of failure to retrieve associated user, just in case.
  return unless defined $user;
  $username = $user->get_value( "username" );
  # $dataset includes all eprints in Inbox, so we limit to $depositor's items only
  return if( $username ne $depositor );


  #skip if no proquest id is associated with the current eprint
  my $pq_id = $eprint->get_value('pq_id');
  if (not defined $pq_id)
  {
    print "EPrint ".$eprint->get_id." doesn't have a proquest id, skipping\n";
    $int_no_proid ++;
```

```
    push ( @array_no_proid, $eprint->get_id );
    return;
  }

  #remove space from proquest id
  $pq_id =~ s/\s//g;

  #attach the PDF to eprint objects and move to live archive
  if ($filemap->{$pq_id} and -e $filemap->{$pq_id} ) #if the file exists
  {
    #create document object, add pdf files to document, attach to eprint object, and move to live
archive
    my $doc = EPrints::DataObj::Document::create( $session, $eprint );
    $doc->add_file( $filemap->{$pq_id}, $pq_id . '.pdf' );
    $doc->set_value( "format", "application/pdf" );
    $doc->commit();
    print "Adding Document to EPrint ", $eprint->get_id, "\n";
    $eprint->move_to_archive;
    print "Eprint ".$eprint->get_id." moved to archive.\n";
    $int_live ++;
  }
  else
  {
    #move the metadata-only eprints to live as well
    print "Proquest ID \\$pq_id\\ (EPrint ", $eprint->get_id, ") does not have a file associated
with it\n";
    $eprint->move_to_archive;
    print "Eprint ".$eprint->get_id." moved to archive without document attached.\n";
    $int_no_doc ++;
    push ( @array_no_doc, $eprint->get_id );
  }
}

#Recursively traverse the directory, find all PDF files.
sub load_filemap
{
  my ($directory) = @_;

  foreach my $filename (<$directory/*>)
  {
    if (-d $filename)
    {
      load_filemap($filename);
    }
    #catch the file name ending in .pdf
    elsif ($filename =~ m/([^\/]*)\.pdf$/i)
```

```
    {
      my $pq_id = $1;
      #add pq_id => filename pair to filemap hash table
      $filemap->{$pq_id} = $filename;
    }
  }
}
```