Voyager Scripts' Guidelines

This document gives a detailed explanation of the scripts and how to work through these scripts and use them effectively. Although these scripts are used to extract data from the Voyager, one could also use it as a guideline to also extract data from other ILS. Sample scripts of other ILS would be available soon.

Voyager – University of Rochester

Introduction

As described in this manual, data needs to be extracted from a commercial ILS and made available to the OAI Toolkit, and this is accomplished via the ILS Connector. This section will detail how the University of Rochester accomplishes this task using the Voyager ILS.

This ILS Connector can be used to produce a bulk export of records or to incrementally extract updated records based on date ranges. Voyager has a tool called marcexport that can be used to export updated and newly created records. This section will describe the process of automating marcexport and moving the MARC records to your OAI server location. The only requirement, besides Voyager, is to have OpenSSL and OpenSSH installed on both your Voyager server and OAI server.

Note that this section will refer to both Voyager (capital "V") and voyager (lower-case "v"). Voyager with a capital "V" refers to the entire Voyager ILS. Lower-case "v" voyager refers to the Unix user, "voyager".

This document represents only one way of automating the MARC export process to your OAI server. There are other ways to accomplish the same thing and it will be ultimately up to you to decide which method is the best for your institution. We recommend that you try to use ILS supported mechanism(s) to accomplish this instead of direct database calls.

Overview

The process of exporting MARC records and moving them is controlled by a shell script called XcExport. XcExport will:

1. Get the current date to use as part of the filename for the exported records file
2. Launch Pmarcexport (marcexport's executable file) with the proper parameters
3. Copy the exported records file to the OAI server
4. Compress the exported records file and move it to an archive directory
5. Remove the original exported records file
6. Check to see if any deleted records need to be moved to the OAI server
7. Will export authority records once a week in a separate process

Note that we have XcExport set up to run once a day as a cron job. Since it only runs once a day, we use the today-0 option as the date range.

Parameters for Marcexport

Voyager's marcexport extracts the records from the database and puts them into a single file with a .mrc extension. This section will describe the parameters we use. There are different options available for this program, which are detailed in Voyager's *Technical User's Guide* for the version of Voyager that you're using.

The entire command is:

```
/m1/voyager/rochdb/sbin/Pmarcexport
-o/m1/voyager/local/bin/XcMarcExport/updates$FILEDATE.mrc -rB -mB -ttoday-0
-aM
```

Pmarcexport is the name of the executable file (not to be confused with marcexport, which is the name of the process). Note that the generic xxxdb is used only for documentation purposes, and you would replace xxxdb with whatever your institution uses as its Voyager instance.

–o	Output - the first option (–o) is for the output filename. This is optional; if not specified the output file will be `marc.exp.yyyymmdd.hhmm` with yyyy = year, mm = month, dd = day, hh = hour and mm = minutes that the script is run. It will also be placed in `/m1/voyager/xxxdb/rpt`. This default name is a little cumbersome, so we opt to give it our own name. $FILEDATE is a variable created by XcExport which is the current date in a nicer format. Also, by using the –o option, we get to put the output file into our current working directory.
-r	Record type - option (-r) is the record type. In this example, we are exporting bibliographic records, so we use the B parameter with the –r option (–rB).
-m	The –m or mode option lets you pick which records you want to export. We use the B parameter for both created and updated records. That means that any records created or updated for the date range passed to marcexport will be exported. If you just wanted updated records, you could the U parameter. If you wanted newly created records only, you would use the C parameter. We want both, so we use the B parameter (-mB).
-t	The –t option is the target date range. We use the today-0 parameter, which means export any record either updated or created on the same day marcexport is run. You can use a date range instead of the today-0 parameter. The date range must be in this format: `yyyy-mm-dd:yyyy-mm-dd`
-a	The last parameter (-a) is character mapping. We use the M option, which is MARC21 MARC-8.

As mentioned in the Overview, the XcExport script actually launches Pmarcexport. XcExport is run once a day at 11:30 PM as a cron job. This will ensure we catch all records created and updated for the entire day.

As a test, run Pmarcexport with the settings shown below to make sure it works properly:

```
/m1/voyager/xxxdb/sbin/Pmarcexport -oupdates.mrc -rB -mB -ttoday-0 -aM
```

XcExport

XcExport is a simple shell script that runs the entire export process. The script is show below.

```
#!/bin/sh
#
# It shouldn't be necessary since we use the full path to executables, but feel free to uncomment
# the PATH statements below and add any relevant path information for your server
#PATH=
#export PATH
#################################################################
#
# MARC record export script, by Ralph Arbelo 4/18/07
# Last edited: 11/11/08
#
# XcExport runs Pmarcexport with the following parameters:
# rG - bibliographic & authority records
# mU - Updated records
# ttoday-0 Records updated today
# aM Assign character mapping, MARC-8,21
#
# The output file is then moved to another server.
#
#################################################################


# Create start date
# Four digit year format
YEAR="`date +%Y`"

# Two digit day format
DAY="`date +%d`"

# Two digit month format
MONTH="`date +%m`"

echo $MONTH
# Put date in format that Pmarcexport expects (YYYY-MM-DD)
FILEDATE=$YEAR-$MONTH-$DAY

# Run MarcExport
/m1/voyager/rochdb/sbin/Pmarcexport -o/m1/voyager/local/bin/XcMarcExport/updates$FILEDATE.mrc -
rG -mB -i -ttoday-0 -aM
```

```
echo Moving file to  XcVoyOAI

/usr/local/bin/scp /m1/voyager/local/bin/XcMarcExport/updates$FILEDATE.mrc
voyager@xcvoyoai:/import/marc_extract

# Zip updates$FILEDATE
echo zipping updates$FILEDATE.mrc
/bin/gzip /m1/voyager/local/bin/XcMarcExport/updates$FILEDATE.mrc

echo Move file to archive directory
mv /m1/voyager/local/bin/XcMarcExport/updates$FILEDATE.mrc.gz
/m1/voyager/local/bin/XcMarcExport/XcArchive/.

echo Finished with record export
echo
echo Checking for a new deleted records file. If exists then copy it to the OAI server

#Check to see if new deleted file exists, if so then move it

if [ -f `find /export/home/voyager/Archive/ -mtime -4 -name "*.tgz"` ] ; then
        /usr/local/bin/scp `find /export/home/voyager/Archive/ -mtime -4 -name "*.tgz"`
voyager@xcvoyoai:/import/deleted_reco
rds
        echo files moved
else
        echo No new file today
fi

# Authority record export
# This section will export authority records once a week (on Monday)

# This will look for Mon and if it is Monday, will put it in the file
date | grep Mon >> /m1/voyager/local/bin/XcMarcExport/tempfile

# If testfile is an empty file, then do nothing. Otherwise, run the export and move them to the OAI server
if [ -s /m1/voyager/local/bin/XcMarcExport/tempfile ] ; then
        echo "It's Monday, run the authority record export"
        /m1/voyager/rochdb/sbin/Pmarcexport
-o/m1/voyager/local/bin/XcMarcExport/auth_records$FILEDATE.mrc -rA -mB -aM -ttoda
y-7
        # Move the extract to the OAI server
        /usr/local/bin/scp /m1/voyager/local/bin/XcMarcExport/auth_records$FILEDATE.mrc
voyager@xcdevvoyoai:/import/marc_extr
act
        # Gzip and move to archive directory
        /bin/gzip /m1/voyager/local/bin/XcMarcExport/auth_records$FILEDATE.mrc
        mv /m1/voyager/local/bin/XcMarcExport/auth_records$FILEDATE.mrc.gz
/m1/voyager/local/bin/XcMarcExport/XcArchive/.
```

```
        echo "Finished moving authority record extract!"
else
        echo "It's not Monday, nothing to do"
fi
rm /m1/voyager/local/bin/XcMarcExport/tempfile

echo XcExport finished!
echo "- - - - - - - - - - - - - - - - - - - - - - - - - "
```

The PATH environmental variable shouldn't be necessary since the script uses full paths to all files. However, having PATH information doesn't hurt and if the script isn't working it's a good first step to try. Please note the path to some of your executable files like scp and tar may be different on your server. Please customize the script accordingly for your environment. .

XcExport creates a variable called FILENAME, which is the current date in a nice format (yyyymmm-dd). Pmarcexport is executed and produces an output file called updates$FILENAME.mrc (for example: updates2008-05-21.mrc).

The output file is then copied to the OAI server via secure copy (details on setting this up are in the next section). Since a copy of the file is moved, we need to do something with the original. Our XcExport script zips the output file (using gzip), copies it to a directory called XcArchive then deletes the original output file.

XcExport will also check for any new deleted/replaced records. We have a routine set up where once a week any deleted/replaced records are combined in a tgz file (see the section on deleted/replaced records for more details). Each time XcExport runs it checks for a recent version (4 days or less old). If it finds one, then it is copied to the OAI server.

Finally, XcExport will extract and move Authority records. The records extracted in the first part of the script are Bib and Holding records. Unfortunately, Pmarcexport doesn't allow the extraction of all three record types at the same time.

We use the same parameters with Pmarcexport except for two:

- -rA
- -ttoday-7

The "A" part of –rA tells Pmarcexport to export Authority records. The –ttoday-7 means export Authority records from the last seven days. Feel free to adjust the time interval to what best suits your inisitution.

The XcExport script checks to see what day of the week it is. If it's Monday, the script will run Pmarcexport and copy the files to the OAI server. Like the Bib and Holding records, the Authority record extract will then be gzipped and moved to an archive directory. Tuesday through Thursday the Authority record section of XcExport will be ignored.

The XcExport script is owned by the user voyager and the group endeavor. It also needs to have execute privileges as shown below.

```
-rwxr--r--   1 voyager  endeavor    1806 May 23 10:25 XcExport
```

The script can live anywhere as long as voyager can write to the directory to create the output file. We like to keep customized scripts that interact with voyager in the /m1 directory structure.
If you want to have this script run as a cron job (which is recommended), it should to be run from voyager's crontab. Schedule it at the end of the day when no one will be updating or adding records to ensure that the script gets them all. XcExport only takes about a minute to run, so it can be scheduled close to midnight. For troubleshooting purposes, it is also recommended that you have a log file for the XcExport cron job. The entire crontab entry is show below:

```
05 23 * * 1-5 /m1/voyager/local/bin/XcMarcExport/XcExport >
/m1/voyager/local/cronlog/XcExport.txt 2>&1
```

Automating a Secure File Transfer between Servers

After marcexport creates an output file, we need to move it to the OAI server. There are a few different ways to accomplish this. We like using Secure Copy (or SCP) because it's secure, comes with OpenSSH and can be automated. While the output file could be manually copied via SCP, you run the risk of someone forgetting and having your XC repository become out of date. Automating the process takes care of that risk.
Automating SCP requires setting up a public/private key authentication. Please read up on OpenSSH public key authentication to make sure you understand the process and security ramifications. You can find more information at http://sial.org/howto/openssh/publickey-auth/.
Both your Voyager server and the OAI server need to have OpenSSL and OpenSSH installed and have sshd (the SSH daemon) running. You may already have both of these installed. To check, enter the following command:

```
ssh –Version
```

You should see this response:

```
OpenSSH_4.6p1, OpenSSL 0.9.8e 23 Feb 2007
```

The version isn't really important, but this might be a good time to update them if they're really old.
Since the output file is owned by voyager, we have voyager SCP the file over. We also have created a voyager user on the OAI server. While you don't have to do it this way, you'll have permission issues to deal with if you select to use different users. Do not set up root to root trust between servers!
To enable public/private key authentication, find the sshd_config file and open it with your favorite text editor. Make the following changes on both servers:

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys
```

On your Voyager server, as the voyager user, generate the key pair. The keys should reside in the .ssh directory in voyager's home directory (/export/home/voyager in Unix, /home/voyager for Linux). Also,

leave the passphrase blank, otherwise you'll be prompted to enter the passphrase whenever you try to use this, thus defeating the purpose of automation.

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key
(/export/home/voyager/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/id_rsa.
Your public key has been saved in .ssh/id_rsa.pub.
The key fingerprint is:
<long string of hexadecimal numbers> voyager@servername
```

The id_rsa key is the private key. Take care to protect it, as anyone who has possession of your private key can gain access to servers with a matching public key! Give it read/write permissions for voyager only. Remove all permissions for group and other.

```
-rw-------   1 voyager  endeavor    1675 Apr 23 12:03 id_rsa
```

On the OAI server, create a .ssh directory in voyager's home directory. Enter that directory and create a file called "authorized_keys" without the quotes. Copy the contents of id_rsa.pub from your Voyager server. It will be one line that looks like this:

```
ssh-rsa <very long string of number & letter> voyager@servername
```

Set the permissions of the .ssh directory to 700 and 600 to authorized_keys as follows:

```
chmod 700 .ssh
chmod 600 authorized_keys
```

That's it. To test it, try to SCP a file from your Voyager server.

```
touch testfile
scp tesfile voyager@oaiserver:/export/home/voyager/
```

If it's working properly, the file will get copied over without having to enter a password. Note that for the very first time you try this, you may be asked to accept the RSA key. This will only be asked one time. Once you accept it, then you won't have to do it again. If that happens, try it again. The file should get copied over without any user interaction.

At this point, you could have the MARC record file copied to Voyager's home directory on the OAI server. In our XcExport script, however, we SCP the file to a directory where the OAI server expects it. If you want to do the same it's just a matter of setting the proper directory ownership. We put the Voyager user in the same group as the owner of the directory we want to SCP the file. There are other ways to accomplish this as well.

The last part of XcExport moves the deleted/replaced records to the OAI server. The deleted/replaced records are actually generated by another script called del_rep_archive as described below.

Deleted and Replaced Records

We need to know which records have been deleted or replaced in your ILS so we can do the same to the Xc repository. Voyager keeps track of them in six separate files located in /m1/voyager/xxxdb/rpt:

- delete.item
- deleted.auth.marc
- deleted.bib.marc
- deleted.mfhd.marc
- discard.bib.marc
- replace.auth.marc
- replace.bib.marc

The process to move them is simple; just stop Voyager, move them to another directory and start Voyager again. Once Voyager starts back up it will recreate those files (which is why it's important to stop Voyager before moving them). This is something you may be already doing as part of your routine maintenance for Voyager (we run this once a week). Depending on the volume of records your library processes, you can adjust yours to run more or less frequently.

The XcExport script will look for

Del_rep_archive

```
#!/bin/sh
#
# Archive deleted and replaced marc files
# Updated 11/13/08
#
# This script will move a list of Voyager reports to Voyager's home directory.
#
# Voyager must be stopped in order to move these files
#
# Create start date
# Four digit year format
YEAR="`date +%Y`"

# Two digit day format
DAY="`date +%d`"

# Three letter month format
MONTH="`date +%b`"
FILEDATE=$YEAR-$MONTH-$DAY

# Stopping Voyager
/etc/init.d/voyager stop
echo Moving Files....
mv /m1/voyager/rochdb/rpt/delete.item /export/home/voyager/Archive/delete.item$FILEDATE
mv /m1/voyager/rochdb/rpt/deleted.auth.marc
/export/home/voyager/Archive/deleted.auth.marc$FILEDATE
mv /m1/voyager/rochdb/rpt/deleted.bib.marc
/export/home/voyager/Archive/deleted.bib.marc$FILEDATE
mv /m1/voyager/rochdb/rpt/deleted.mfhd.marc
/export/home/voyager/Archive/deleted.mfhd.marc$FILEDATE
```

```
mv /m1/voyager/rochdb/rpt/discard.bib.marc
/export/home/voyager/Archive/discard.bib.marc$FILEDATE
mv /m1/voyager/rochdb/rpt/replace.auth.marc
/export/home/voyager/Archive/replace.auth.marc$FILEDATE
mv /m1/voyager/rochdb/rpt/replace.bib.marc
/export/home/voyager/Archive/replace.bib.marc$FILEDATE
echo Finished moving files

# Starting Voyager
/etc/init.d/voyager start

# Move to Archive directory
cd /export/home/voyager/Archive

echo Tar and zipping files
/usr/local/bin/tar cvzf deleted.$FILEDATE.tgz *marc$FILEDATE

echo Changing ownership
chown voyager:endeavor *$FILEDATE.tgz

echo Archive_marc finished
```

Just like the XcExport, we place the current date into the file name (this will be important for another script that checks to see if the file arrived). The script then moves the files to an archive directory in Voyager's home directory. Once the files are there, we tar them in a single file archive (except for the item records which we don't use in our Xc implementation) and compress it with gzip. Note if you want to tar and gzip in one command, you have to use GNU tar. The version of tar that ships with Solaris doesn't understand the –z option (zipping).

The script does a cd into the archive directory instead of using the full (or absolute) path to the files in the tar command. If you use the absolute path in the tar command, that path will be recreated when they're unzipped and utarred on the OAI server.

This script is run as root (unlike the XcExport script which is run as voyager) since we have to stop and start Voyager. To make sure the ownership of the tgz file is correct, the script will set the owner to voyager and the group to endeavor. If this file is owed by root the SCP will fail.

The file is moved to the OAI server by the XcExport script. It's done this way for a couple of reasons. First, we don't allow root SSH access on our servers for security reasons. While you can use su to run SCP as Voyager (su – voyager –c) in a script, you get a new shell and lose the $FILEDATE variable. Instead of creating a new script just to move this file over, we added a section in the XcExport script to do it. Since that script uses the same variables, it's a perfect fit.

This is the section of code that moves the deleted.$FILEDATE.tgz:

```
if [ -f `find /export/home/voyager/Archive/ -mtime -3 -name "*.tgz"` ] ; then
        /usr/local/bin/scp `find /export/home/voyager/Archive/ -mtime -4 -name "*.tgz"`
voyager@xcdevvoyoai:/import/deleted_records
else
```

```
        echo No new file today
fi
```

It does a find in the /export/home/voyager/Archive directory for a file that was created in the last three days and ends with .tgz. If such a file exists, then it is moved to the OAI server. If it doesn't exist then the script will just echo "No new file today" in the log.

Why does this file use the find command instead of just moving the file directly? Timing. The del_rep_archive script runs on Saturday morning while the XcExport script runs Monday through Friday night. The deleted.$FILEDATE.tgz file will have Saturday's date in the file name. When the XcExport script runs on Monday, it will have Monday's date in the $FILEDATE variable. If the script checks for the file using the $FILEDATE variable, it will always fail. The find command will always come back with the latest version, but skip over any older files that may be in that directory.

Another way around this is to just have your XcExport script run on Saturdays or to have your del_rep_archive script run on a weekday. Or you could have a directory that contains only the latest deleted.$FILEDATE.tgz file and always move whatever is in it. Once it's moved to the OAI server, you would then have to move that file to another directory. There are many ways to accomplish what these scripts do.

Monitoring

It's important to set up some sort of mechanism in place to monitor both XcExport and archive_marc to make sure the files are actually getting to the OAI server. Since both of those files are scheduled via cron, you should have a log file to record what's going on when the script is running. Here (again) is the crontab entry for XcExport:

```
05 23 * * 1-5 /m1/voyager/local/bin/XcMarcExport/XcExport >
/m1/voyager/local/cronlog/XcExport.txt 2>&1
```

The log file, XcExport.txt, gets overwritten every day. You could choose to append to this file instead of overwriting by using ">>" instead of " >". We felt this wasn't necessary because of the scripts that we have on the OAI server (which we'll get to in the next paragraph). While the log files are nice, it still makes checking to see if the files actually arrived a manual process.

Instead of having to check two log files everday, we wrote two scripts that will check to see the files arrived. If they didn't then the OAI server will send you an email (both these scripts run from the OAI server). This is the script that checks for XcExport's file:

xcexport_monitor_script

```
#!/bin/sh
#
# This script will check to see if the BIB and Authority file extract arrrived
#Create date formatYEAR="`date +%Y`"

# Two digit day format
DAY="`date +%d`"

# Three letter month format
MONTH="`date +%b`"
FILEDATE=$YEAR-$MONTH-$DAY

echo $FILEDATE
if [ -f /import/marc_extract/updates$FILEDATE.mrc ] ; then
echo XcExport file arrived ;
else
cat <<EOF | mailx -s "Bib & Auth Records Did Not Arrive" rarbelo@library.rochester.edu
EOF
Fi
```

We use the familiar $FILEDATE variable for consistency and simplicity. The script then does a file check to see if that file exists. If it does, then it will echo “XcExport file arrived” into the log file. If the file hasn’t arrived then the script will send out an email.

The script to check for the deleted/replaced file export works a little differently. Since its created three days before it is actually moved, we can’t use the $FILEDATE method like the xcexport_monitor_script. We instead have to use the find command in the same way the XcExport script does:

delrep_monitor_script

```
#!/bin/sh
find /import/deleted_records -mtime -4 -name "*.tgz" > /import/deleted_records/TESTFILE
if [ -s /import/deleted_records/TESTFILE ] ; then
echo Deleted file arrived ;
time
else
echo "Deleted records did not arrive" >> /import/deleted_records/mailfile
date
echo "Deleted records did not arrive"
mailx -s "Deleted Records Did Not Arrive"
rarbelo@library.rochester.edu,shreyanshv@library.rochester.edu < /import/deleted_re
cords/mailfile
fi
rm /import/deleted_records/TESTFILE
rm /import/deleted_records/mailfile
```

Here’s the script to monitor the Authority record import:

Auth_Monitor_Script

```
#!/bin/sh
```

```
#Create date format
YEAR="`date +%Y`"

# Two digit day format
DAY="`date +%d`"

# Two digit month format
MONTH="`date +%m`"
FILEDATE=$YEAR-$MONTH-$DAY

if [ -f /import/marc_extract/auth_records$FILEDATE.mrc ] ; then
date
echo Bib Auth file arrived ;
else
date
cat <<EOF | mailx -s "Authority Records Did Not Arrive" rarbelo@library.rochester.edu, shreyanshv@library.rochester.edu
EOF
fi
```

Timing is important in order for these scripts to work properly. We schedule these scripts to run 15 minutes after XcExport. That gives plenty of time for the exported files to arrive (it usually only takes a minute or two). Also, make sure that they're scheduled to run on the same days as the scripts they are monitoring. For us, delrec_monitor_script and Auth_Monitor_Script runs once a week on Monday while xcexport_monitor_script runs Monday thought Friday.

Finally, make sure that both your Voyager and OAI server are using NTP or some other time synchronization mechanism. The system times for both servers need to be in sync with each other. Otherwise, you might end up with a monitoring script going off before the script that's supposed to generate and move the file.

These scripts are provided as a convenience for keeping your Xc repository up to date. As mentioned before, there are many ways to accomplish what these scripts do. It's ultimately up to you and your institution to determine the best way.

Putting it all Together

Now that all the elements are working, put your XcExport and del_rep_archive (if you don't' already have one) scripts in place. If XcExport doesn't work correctly, comment out everything except for the marcexport. If that works, then enable the SCP portion. Keep going until you find the piece that isn't working. Run it manually a few times to be sure it works properly before scheduling it as a cron job.