Using texql and texexport to export data from EMu

From Larry Gall to Pete Herbst November 2020

The following shows the syntax of the texexport command. You feed the texexport command a bunch of options followed by the name of the EMu table (=module or database) for which you want to dump data. For example: eparties, ecatalogue.

bash-5.0$ texexport -?

Usage: texexport [-R] [-V] [-ccollist] [-fstyle] [-ssortlist] \

[-ttype] [-mmodifiers] [-nr] -a|-kfile|-lkeylist|-ofile dbname

Where

-ccollist is list of columns to export (col[:n],col[:n]...)

-fstyle is style of output (see below)

-ssortlist is list of columns for sorting (col[:a|d],col[:a|d]...)

-ttype is type of exporting (forms or schema)

-mmodifiers defines modifier options for selected style

-n means notify upon completion

-r means remove file at completion

You must specify one of:

-a export all records from dbname table

-kfile file of keys of rows in dbname table

-lkeylist comma-separated list of keys of rows in dbname table

-ofile file of offsets of rows in dbname table

Formats include:

ids lines of the format colid:fieldno=data

idsbrief as for ids but with empty lines omitted

options -ml print Library items as single value

-mwn print col name in exactly n characters

texpressdata KE Texpress internal data file format

texql KE Texql insert statements

delimited character-separated columns

options -mdch use ch as text delimiter (dflt ")

-msch use ch as column separator (dflt tab)

-mc include column names on first line

-ml print Library items as single value

fixedwidth fixed width (blank-padded) output

format column output defined by format string

formatbrief as for format but with empty lines omitted

options -mfstr use str as the format string

-ml print Library items as single value

The texexport command is flexible and very fast and is great for command line operations especially (a) dumping data out in various textual formats, (b) limiting that dump to certain database columns, and (c) dumping subsets of records (from one, to some, to all records in a database).

The output format that I use the most is "idsbrief" which means you get data that are "formatted, with column ids, omitting blank data fields." Using "idsbrief" will display one line for an atomic field, and one or more lines for a table; each line will be prefixed with the name of the column then a colon and then its index (1 for an atomic field, up to N for N rows of a table).

The next most common format I use is "delimited" which means that all the fields for one record are printed on a single line, with delimiters (and where you can control the delimiters).

Here is a typical texexport dump. I've used the "-f" option (with idsbrief) to print each column on its own line, and the "-l" option (with the internal record number 5000) to show me what is in that one record in the ecatalogue table:

bash-5.0$ texexport -fidsbrief -l5000 ecatalogue

rownum=1

irn:1=5000

SummaryData:1=ANT.005214: Net sinkers (4). From the region of Oneonta, Otsego County, N. Y. Colls and donors L. B. Capron ('13) and Mr. Jaeger. New York. Otsego County.

CatDepartment:1=Anthropology

CatFullNumber:1=ANT.005214

CatPrefix:1=ANT

CatNumber:1=5214

CatSpecimenCount:1=4

CatOriginalCount:1=4

IdeCat:1=Net sinkers (4). From the region of Oneonta, Otsego County, N. Y. Colls and donors L. B. Capron ('13) and Mr. Jaeger.

DetFiledAs:1=Yes

IdeMostRecentFiledAs:1=Yes

IdeMostRecentCat:1=Net sinkers (4). From the region of Oneonta, Otsego County, N. Y. Colls and donors L. B. Capron ('13) and Mr. Jaeger.

IdeFiledAsFiledAs:1=Yes

IdeFiledAsCat:1=Net sinkers (4). From the region of Oneonta, Otsego County, N. Y. Colls and donors L. B. Capron ('13) and Mr. Jaeger.

CatSiteRef:1=36213

CatSiteRefLocal:1=36213

CatCountryLocal:1=USA

CatProvinceStateTerritoryLocal:1=New York

CatDistrictCountyShireLocal:1=Otsego County

CatNearestNamedPlaceLocal:1=Oneonta

CatLatitudeDecLocal:1=42.45724

CatLongitudeDecLocal:1=-75.06166

CatDatumLocal:1=WGS84

CatDeterminationMethodLocal:1=digital resource

CatRadiusNumericLocal:1=2426.89075

CatContinentLocal:1=North America

DesMaterial:1=stone

DesMaterial:2=wood

CulCulture:1=North American : Eastern Woodland

AntFunctionalGroup:1=tools & manufacturing equipment

{I've snipped out quite a few more lines of output here!}

AdmDateInserted:1=2005

AdmDateInserted:2=5

AdmDateInserted:3=9

AdmTimeInserted:1=16

AdmTimeInserted:2=27

AdmTimeInserted:3=59.000

AdmModifiedBy:1=emu

AdmDateModified:1=2020

AdmDateModified:2=06

AdmDateModified:3=15

AdmTimeModified:1=23

AdmTimeModified:2=17

AdmTimeModified:3=30.000

SecCanDisplay:1=Group Default

SecCanEdit:1=Group Default

SecCanDelete:1=Group Default

SecDepartment:1=Anthropology

###

When you use the idsbrief format, each record that is printed gets wrapped with a header line that is "rownum=N" (where N is the number of records dumped) and a trailer line that is "###". This will, for example, let you figure out in scripts where a given record starts and stops.

From the above, on each line you see the column name followed by the index followed by the actual data. For example, SummaryData is a field present in each module at each EMu institution, and it is atomic, and hence you get "SummaryData:1=blahblah". The field DesMaterial is one that is specific to the Peabody (and/or perhaps other folks) that describes what material(s) an artifact consists of, and it is a table, and on this record there happen to be two values recorded in the table… stone and wood. So you get "DesMaterial:1=stone" for the first and "DesMaterial:2=wood" for the second. You can also see that the various administrative fields for the timestamps and datestamps that EMu keeps track of for each record are expressed in tabular manner. [Sidebar: in general, when using the idsbrief format, the index value will cleanly reflect the rows in a table; but if you have a text field into which you have manually inserted newlines using the EMu client, then you will also get output rows that represent those newlines, in addition to rows for data per se).

OK, add the Unix egrep command onto the previous texexport command to limit output to lines having either the string "irn" or "Local" in them:

bash-5.0$ texexport -fidsbrief -l5000 ecatalogue | egrep 'irn|Local'

irn:1=5000

CatSiteRefLocal:1=36213

CatCountryLocal:1=USA

CatProvinceStateTerritoryLocal:1=New York

CatDistrictCountyShireLocal:1=Otsego County

CatNearestNamedPlaceLocal:1=Oneonta

CatLatitudeDecLocal:1=42.45724

CatLongitudeDecLocal:1=-75.06166

CatDatumLocal:1=WGS84

CatDeterminationMethodLocal:1=digital resource

CatRadiusNumericLocal:1=2426.89075

CatContinentLocal:1=North America

AccAccessionLotRefLocal0:1=5042

Ok, now instead use the "delimited" format instead of "idsbrief" and print just two fields rather than all field using the "–c" option. My two fields are irn and CatDepartment. The fields in the list get separated by commas:

bash-5.0$ texexport -fdelimited -l5000 -cirn,CatDepartment ecatalogue

5000 "Anthropology"

OK, now change the default column separator character using the "-md" option and the default text separator using the "-ms" option. I've told texexport not to bother with the column separator and use an underscore for the text separator:

bash-5.0$ texexport -fdelimited -l5000 -md -ms\_ -cirn,CatDepartment ecatalogue

5000\_Anthropology

Probably it is now more apparent how to use these two texeport operations to vary the format of the printed output. Once you catch on, then piping that output with other Unix commands will also make sense. Redirecting output to a file is a common task.

OK, the "-l" option was great for printing one record out. But you can use the "-k" option to print out a set of records when you have their irns, or the "-a" option to print out ALL records in a given EMu table (only use the "-a" opption once you feel quite comfortable with texexport). Here is the contents of a file called junk.txt which has just two lines in it, one line for each of two different irns:

bash-5.0$ cat junk.txt

300

1425005

OK, feed it to texexport:

bash-5.0$ texexport -fdelimited –kjunk.txt -md -ms\_ -cirn,CatDepartment ecatalogue

300\_Anthropology

1425005\_Entomology

You could also have done the same thing using the "-l" option, since there were only two irns in total, whereas the "-k" option is much preferable when you go beyond small numbers of irns to dump:

bash-5.0$ texexport -fdelimited –l300,1425005 -md -ms\_ -cirn,CatDepartment ecatalogue

300\_Anthropology

1425005\_Entomology

Hopefully this is just enough to get started with texexport. For those familiar with command line scripting and/or Unix piping, it should now be apparent that you can do some very useful things with texexport. You can also combine texexport with other Texpress utilities, such as texql (typically to identify a set of records based on some criteria and then feed the result set of irns into a file for texexport to act on). Or, use texexport simply to build up some suitably crafted output files as raw material that you can then process in much greater detail in some other application, such as Perl, Python, whatever you love.