How to run the xref pipeline

Added by Monika Komorowska, last edited by Magali Ruffier on Jul 18, 2013

1) Prerequisites

- Ensembl API - head code, with API version updated to the current release in progress.
- Exonerate - the default exonerate path can be used for Sanger. If you don't already have it installed, installation intructions can be found at http://www.ebi.ac.uk/~guy/exonerate/
- Enough disk space to store xref files, sequences and mapping files. For human this is about 10Gb (release 68).
- Core databases handed over and ready for xrefs.
- Vega databases handed over.
- ensembl_ontology_[release number]

2) Configuring the system

Edit the xref_config.ini file.

If you need to add a new species, add a new entry in the species section.

Here is an example:

```
[species macaca_mulatta]
taxonomy_id = 9544
aliases = macaque, rhesus, rhesus macaque, rmacaque
source = EntrezGene::MULTI
source = GO::MULTI
source = InterproGO::MULTI
source = Interpro::MULTI
source = RefSeq_dna::MULTI-vertebrate_mammalian
source = RefSeq_peptide::MULTI-vertebrate_mammalian
source = Uniprot/SPTREMBL::MULTI
source = Uniprot/SWISSPROT::MULTI
source = Uniprot/SWISSPROT::DIRECT
source = UniProt::protein_id
source = Uniprot::EMBL
source = UniParc::MULTI
source = RFAM::MULTI
source = miRBase::MULTI
source = ArrayExpress::MULTI

[species xxxx] and taxonomy_id must be present.
```

For RFAM and ArrayExpress, the pipeline will need to access your core database.
By default, it will expect to find it on one of the staging servers.
If the database is on an alternative server, this needs to be specified in the source configuration.
Here is an example:

```
[source ArrayExpress::MULTI]
# Used by all ensembl species
name            = ArrayExpress
download        = Y
order           = 50
priority        = 1
prio_descr      =
parser          = ArrayExpressParser
release_uri     =
#data_uri        = script:wget=>http://www.ebi.ac.uk/gxa/dataExport/organisms,
data_uri        = script:wget=>http://www.ebi.ac.uk/gxa/dataExport/organisms,p
```

In case of a new xref source, add a new entry in the sources section
Here is an example:

```
[source Fantom::mus_musculus]
#Used by mus_muscullus
name = Fantom
download = Y
order = 100
priority = 1
prio_descr =
parser = FantomParser
release_uri =
data_uri=ftp://fantom.gsc.riken.jp/DDBJ_fantom3_HTC_accession.txt.gz
```

name: The name you want to call the external database. You must also add this to the core databases

download: Y if the data needs to be obtained online (i.e. not a local file) N if you are getting the data from a file.

order: The order in which the source should be parsed. 1 beinging the first.

priority: This is for sources where we get the data from multiple places i.e. HGNC. For most sources just set this to 1.

prio_desc: Only used for priority sources. And sets a description to give a way to diffentiate them and track which is which.

dependent_on: Comma separated list of sources which must be loaded first. Note that if species does not have xrefs from a master source specified in this list than the dependency is ignored.

parser: Which parser to use.

release_uri: a uri to get the release information from. The parser should handle this.

data_uri: Explains how and where to get the data from. There can be multiple lines of this. The uri can get data via several methods and here is the list and a brief explaination.

- **ftp:** Get the file via ftp

- **script:** Passes arguments to the parser. This might be things like a database to connect to to run some sql to get the data..
- **\file\:** The name with full path of the file to be parsed.
- **http:** To get data via an external webpage/cgi script.

Update the release version for any databases which are used by parsers (e.g. ccds or ensembl_ontology).

3) Updating the ccds database

Because the stable ids may have changed in the core database we need to update these in the ccds databases.

For release 68 only human and mouse had a ccds database.

The script to run is store_ccds_xrefs.pl and is in the directory ensembl-personal/genebuilders/ccds/scripts.

Submit the job to the farm with 500Mb memory requirement.

Example command for human:

```
bsub -q normal -M 500000 -R'select[mem>500] rusage[mem=500]' -o ccds.out -e cc
perl ~/ensembl-personal/genebuilders/ccds/scripts/store_ccds_xrefs.pl -ccds_db
-ccds_host ens-livemirror -ccds_user ensadmin -ccds_pass xxx -dbname homo_sapi
-host ens-staging1 -port 3306 -user ensro -verbose -species human -path GRCh37
```

If a ccds database is being used, the xref_config.ini needs to be updated to point to the correct, latest up-to-date ccds database.

```
[source CCDS::homo_sapiens]
# Used by homo_sapiens
name            = CCDS
download        = Y
order           = 10
priority        = 1
prio_descr      =
parser          = CCDSParser
release_uri     =
data_uri        = script:host=>ens-livemirror,dbname=>ccds_human_71,tran_name=
[source HGNC::homo_sapiens#01]
# Used by homo_sapiens
name            = HGNC
download        = Y
order           = 30
priority        = 2
prio_descr      = ccds
parser          = HGNC_CCDSParser
release_uri     =
data_uri        = script:wget=>http://www.genenames.org/cgi-bin/hgnc_downloads
[source RefSeq_dna::homo_sapiens]
# Used by homo_sapiens
name            = RefSeq_dna
download        = Y
order           = 25
priority        = 1
prio_descr      = ccds
parser          = RefSeq_CCDSParser
release_uri     =
data_uri        = script:host=>ens-livemirror,dbname=>ccds_human_71,
```

4) Updating LRGs

Good docs can be found here: Importing LRGs into Ensembl

Check out the LRG modules and add them to your perl5lib variable.

Use script ensembl-personal/mk8/lrg/lrg_commands.pl to generate the commands:

```
perl lrg_commands.pl -pass (password for user ensadmin for ens-staging1) -db_v
(version of the human db on ens-staging) -script_path (path to the LRG modules
-out_path (output path for farm job output)
```

5) Running the parsing

Some sources require the downloading of files over HTTP.

If you are firewalled then make sure you have set the HTTP_PROXY environment variable.

For Sanger, the variable should be set to:

```
http://cache.internal.sanger.ac.uk:3128
```

For tcsh shells you should have

```
    setenv http_proxy http://cache.internal.sanger.ac.uk:3128
```

in your ~/.tcshrc file, while for bash-like shell you should have

```
    export http_proxy=http://cache.internal.sanger.ac.uk:3128
```

in your ~/.profile or ~/.bashrc file.

Your perl5lib environment should contain ensembl/modules and ensembl/misc-scripts/xref_mapping

```
setenv PERL5LIB ensembl/modules:ensembl/misc-scripts/xref_mapping
```

cd to where you want the files to be downloaded to or specify option -download_path and run the following:

```
bsub -q normal [-M 1000000 -R'select[mem>1000] rusage[mem=1000]']
-o parse.out -e parse.err perl ~/src/ensembl/misc-scripts/xref_mapping/xref_pa
-user rw -pass password -host ens-research -dbname
ianl_dog_xref_65 -species dog -create -stats -force
```

For human you need to request 1.5GB of memory.

For non-merged species, 500MB is more than enough.

Script options:
-species : which species to start the parsing for
-create : tells the script to create a new database even if one exists already
-stats : gives you statistics about what xrefs have been added for each parser
-force : means no interaction (i.e. for the farm) so it assumes yes to all questions

Explanation of the output:

```
Options: -user rw -pass password -host ens-research
 -dbname ianl_human_xref_65 -species human -stats -create -force
```

Tells us what options were used when the parser script was run.

```
 ----{ XXXX }-------------------------------------------------------------
```

output from the parser XXXX

```
Parsing script:host=>ens-livemirror,dbname=>ccds_human_65,tran_name=>ENST, wit
```

XXXX is being parsed with the XXXXParser (see ensembl/misc-scripts/xref_mapper/XrefParser/XXXXParser.pm for the module).

```
source          xrefs   prim    dep     gdir    tdir    tdir    coord   synony
XXX_transcript  0       0       0       0       33689
XXXX            26451   0       0       0       0       0       0       0
```

So the Parser added 26451 xrefs and 33689 direct xrefs to the transcripts.

Note: we can have more direct xrefs than xrefs as one xref may go to a few transcripts, this is not a problem.

```
==============================================================================
Summary of status
==============================================================================
                          CCDS CCDSParser                OKAY
                        DBASS3 DBASSParser               OKAY
                        DBASS5 DBASSParser               OKAY
                    EntrezGene EntrezGeneParser          OKAY
                            GO GOParser                  OKAY
                            GO InterproGoParser          OKAY
                          HGNC VegaOfficialNameParser    OKAY
                          HGNC HGNC_CCDSParser           OKAY
                          HGNC HGNCParser                OKAY
```

The status for each parser should be "OKAY".

If any of these are not "OKAY" then there has been a problem so look further up in the file to find out why it failed.

If you need to debug the parser you can run parsing with option **-source** for the source processed by the parser

which failed.

If the source is dependent on other sources, you will need to list them all in the **-source** option.

Source line DEPENDENT_ON in xref_config.ini stores a list of sources on which the source being defined is dependent.

If you need to rerun parsing and you don't want to download all the files again, use option **-checkdownload** as this will not download data you already have but will try to get the data you are missing, saving time.

6) Running the mapping

First create a configuration script to tell the mapper program information it needs. Here is an example:

```
############################################################
xref
host=ensembl-host1
port=3306
dbname=human_xref_65
user=rw
password=xxxx
dir=./xref

species=homo_sapiens
host=ensembl-host2
port=3306
dbname=homo_sapiens_core_65_37
user=rw
password=xxxx
dir=./ensembl
pr_host = ensembl-old
pr_user = ro
pr_dbname = homo_sapiens_core_64_37

farm
queue=long
exonerate=/software/ensembl/bin/exonerate-1.4.0
############################################################
```

```
xref
host=ensembl-host1
port=3306
dbname=human_xref_65
user=rw
password=xxxx
```

defines what is needed to connect to the xref database

```
dir=./xref
```

Sets where to dump the xref databases fasta files. Note the directory must exist already.

```
species=homo_sapiens
host=ensembl-host2
```

```
port=3306
dbname=homo_sapiens_core_65_37
user=rw
password=xxxx
```

Defines what is needed to connect to the core database.

```
dir=./ensembl
```

Sets where to dump the core databases fasta files. Note the directory must exist already.

```
pr_host = ensembl-archive
pr_user = ro
pr_dbname = homo_sapiens_core_64_37
```

Normally as part of the xref mapping we check the number of xrefs in the core database to the one in the xref database and

flag any sources that have changed by more than 5%, as this may indicate that we have a problem. By specifying pr_... we are

instructing the comparison to be to another core database. This is normally done when the core database we are updating does

not have a full set of xrefs already and hence the comparison would be useless (this is the case for merge species, such as human, mouse and zebrafish).

```
farm
queue=long
exonerate=/software/ensembl/bin/exonerate-1.4.0
```

Instead of using the default farm queue or exonerate executable we can overwrite these here.

Typically the EBI and Sanger have different queues and other organisations may also differ so this is very useful.

If you're running xrefs for a new species, make sure that you're happy with the precedence of

xref sources for **gene and transcript display xrefs**. The external databases to be used for the display_xrefs

are taken from DisplayXrefs.pm as default (subroutines transcript_display_sources and gene_display_sources).

In case you need to specify different lists, create a [species name].pm module in xref_mapping/XrefMapper/

and overload the subroutines.

A similar approach applies to **gene descriptions**. You can overload sub gene_description_sources (default in DisplayXrefs.pm),

which provides the precedence of xref sources which will be used to populate gene descriptions.

So we are now ready to run the mapping. We need to tell the mapper where the configuration file is (see above).

The mapper is ran twice generally. The first time does all the major work like dumping the fasta files, mapping these files,

reading in the mapping files, and creating all the connections. At this stage a comparison of the xrefs in the core database and new xref database is done.

A typical command line call would be (for human you need to request 1.5GB of memory, for non merged species, you only need to request 500MB):

```
bsub -q normal [-M 1000000 -R'select[mem>1000] rusage[mem=1000]']
-o mapper1.out -e mapper1.err perl xref_mapper.pl -file config_file
```

If you do not have access to a compute farm then :

```
perl xref_mapper.pl -file config_file -nofarm >& mapper1.out
```

(but this will be slow)

If everything looks okay we will then transfer the data by adding -upload to the command line options, i.e. when using the farm

```
bsub -q normal [-M 1000000 -R'select[mem>1000] rusage[mem=1000]']
     -o mapper2.out -e mapper2.err perl xref_mapper.pl
     -file config_file -upload
```

To track the status of the mapping use script xref_tracker.pl:

```
perl ensembl/misc-scripts/xref_mapping/xref_tracker.pl -file config_file
```

To check available statuses login to the db server where your xref db is located and describe table process_status:

```
mysql -h ensembl-host1 -u rw -p xxx
mysql> use human_xref_65;
mysql> desc process_status;
```

Explanation of the output:

```
Options: -file xref_input
running in verbose mode
```

Informs the user how the mapper was run

```
current status is parsing_finished
```

Reports the current status of the xref_database. This is used to work out what to do next.

```
No alt_alleles found for this species.
```

Only for human and mouse do we import the alt_alleles.

```
Dumping xref & Ensembl sequences
Dumping Xref fasta files
Dumping Ensembl Fasta files
53067 Transcripts dumped 41693 Transaltions dumped
```

Reports what files are dumped. If these are already dumped and the option -dumpcheck was used then this will be

report

and if the fasta files already exist they will not be re dumped.

```
Default exonerate method is ExonerateGappedBest1
Default exonerate method overridden for some sources
Will use ExonerateGappedBest5 method for source id 177, RefSeq_mRNA_predicted
Will use ExonerateGappedBest_100_perc_id method for source id 255, Uniprot/SWI
Will use ExonerateGappedBest_100_perc_id method for source id 251, Uniprot/SPT
Will use ExonerateGappedBest_100_perc_id method for source id 250, Uniprot/SPT
Will use ExonerateGappedBest5 method for source id 178, RefSeq_ncRNA
Will use ExonerateGappedBest1 method for source id 184, RefSeq_peptide_predict
Will use ExonerateGappedBest5 method for source id 179, RefSeq_ncRNA_predicted
Will use ExonerateGappedBest5 method for source id 175, RefSeq_mRNA
Will use ExonerateGappedBest1 method for source id 300, miRBase
Will use ExonerateGappedBest1 method for source id 180, RefSeq_peptide
Will use ExonerateGappedBest1 method for source id 216, UniGene
```

Reports the default exonerate method. If the method was overridden for some sources these will be listed together with the non default exonerate methods.

```
Deleting out, err and map files from output dir: /workdir/release_65/zebrafish
Deleting txt and sql files from output dir: /workdir/release_65/zebrafish/ense
LSF job ID for main mapping job: 887287, name ExonerateGappedBest1_1318933449
 481 arrays elements)
LSF job ID for main mapping job: 887288, name ExonerateGappedBest1_1318933451
 253 arrays elements)
LSF job ID for Depend job: 887289 (job array with 1 job)
already processed = 0, processed = 734, errors = 0, empty = 0
```

This is information on the mapping of the fasta files using exonerate. Check that the errors are 0 else one of the mappings went wrong.

If a problem is reported with the jobs in mapping1.err, e.g.:

```
ExonerateGappedBest1_peptide_6_*.map was empty could be okay but if there are
```

it can be caused by running out of disk space. Make sure you provide enough disk space and rerun the mapping.

You have two options:

1) reset the database to the parsing stage and rerun all the mappings

To reset the database use the option -reset_to_parsing_finished

```
xref_mapper.pl -file config_file -reset_to_parsing_finished
```

then redo the mapping

```
xref_mapper.pl -file config_file -dumpcheck
```

Note here we use -dumpcheck to make the program does not dump the fasta files if they are already there,

as this process can take along time and the fasta files will not have changed.

2) just redo those jobs that failed

Run the mapper with the -resubmit_failed_jobs flag

```
xref_mapper.pl -file xref_config -resubmit_failed_jobs
```

Option 2 will be much faster as it will only redo the jobs that failed.

```
Could not find stable id ENSDART00000126968 in table to get the internal id he
 ignoring!!! (for RFAM)
Could not find stable id ENSDART00000121043 in table to get the internal id he
 ignoring!!! (for RFAM)
```

Sometimes external databases will have links to EnsEMBL that are no longer valid, usually due to time delays in the releases of the external database.
Here we can see two of these for RFAM, as long as this number is not too large this is not a problem.

```
The following will be processed as priority xrefs
        Uniprot/SPTREMBL
        ZFIN_ID
```

Priority xrefs are those xrefs where we get the data from more than one place. These will have priorities that tell us which source

is better so the best ones are chosen at this point.

```
Process Pairs
Starting at object_xref of 922335
translation object_xrefs updated:       160
translation object xrefs added: 502
translation object xrefs removed:       727
```

This applies to RefSeq_peptide and RefSeq_mRNA which are considered pairs. If RefSeq_peptide xref was mapped to ensembl protein

and it's corresponding RefSeq_mRNA was mapped to the transcript which produces the protein - this is treated as the best match.

Remaining RefSeq_peptide matches are deleted if this best match exists.

We also match the RefSeq_peptide xref to a translation if it's transcript was sequence matched to the corresponding RefSeq_mRNA.

```
Writing InterPro
246386 already existed
Wrote 0 interpro table entries
    including 51399 object xrefs,
    and 51399 go xrefs
```

We create extra mapping using the InterPro table and these are the stats for this.

```
ZFIN_ID is associated with both Transcript and Translation object types
Therefore moving all associations from Translation to Transcript
```

If a particular source in this example ZFIN_ID is linked to more than one of Gene, Transcript or Translation

then all are moved to the highest level. Gene being the highest and Translation the lowest.

```
DBASS3 moved to Gene level.
DBASS5 moved to Gene level.
```

Some sources are considered to belong to genes but may be mapped to transcripts or translations so we move these now to the gene.

```
For gene ENSDARG00000001832 we have mutiple ZFIN_ID's
        Keeping the best one si:ch1073-403i13.1
        removing zgc:113912 from gene
        removing zgc:103599 from gene
Multiple best ZFIN_ID's using vega to find the most common for ENSDARG00000057
        lratb  (chosen as first)
        wu:fj89a05  (left as ZFIN_ID reference but not gene symbol)
```

For some sources (HGNC in human, MGI in mouse and ZFIN_ID in zebrafish) we only want to have one reference

per gene so using things like their priorities, %id mapping values etc. we try to find the best one and remove the others.

If we cannot find a best one then all are kept.

```
Warning Could not find id for ZFN156 came from ZFN156-001 for HGNC
Warning Could not find id for ZFN156 came from ZFN156-003 for HGNC
Warning Could not find id for ZFN156 came from ZFN156-002 for HGNC
Warning Could not find id for ZFN156 came from ZFN156-005 for HGNC
Warning Could not find id for ZFN156 came from ZFN156-006 for HGNC
```

In case you see a similar message this indicates that Vega and HGNC names are out of synch.
See JIRA ticket http://www.ebi.ac.uk/panda/jira/browse/ENSCORESW-227 for more detail.

```
WARNING: Clone_based_ensembl_gene has decreased by -5 % was 7652 now 7194
WARNING: Clone_based_ensembl_transcript has decreased by -8 % was 8260 now 755
WARNING: xrefs miRBase_gene_name are not in the new database but are in the ol
WARNING: xrefs OTTG are not in the new database but are in the old???
WARNING: xrefs OTTT are not in the new database but are in the old???
WARNING: RefSeq_ncRNA has increased by 5% was 644 now 677
WARNING: xrefs RFAM_gene_name are not in the new database but are in the old??
WARNING: xrefs shares_CDS_and_UTR_with_OTTT are not in the new database but ar
 in the old???
WARNING: xrefs Vega_translation are not in the new database but are in the old
WARNING: ZFIN_ID_curated_transcript_notransfer has 9748 xrefs in the new datab
 but NONE in the old
```

Generally its ok when the count of object xrefs for these sources go down:

```
Clone_based_ensembl_%
Clone_based_vega_%
```

This means a better xref to name the gene (stored in display_xref_id in gene) was found (such as HGNC for instance).

An increase in object xrefs is what we want to see, but if its too big, it might be caused by a bug (anything up to 20% should be ok).

A decrease in any predicted source (e.g. RefSeq_mRNA_predicted) is also ok as more experimental data becomes available.

Some xrefs are not loaded into core using the xref pipeline.Theyre done during Ensembl Havana merge, such as:

```
OTTG
OTTT
shares_CDS_and_UTR_with_OTTT
shares_CDS_with_ENST
shares_CDS_with_OTTT
UCSC
xrefs Vega_transcript
xrefs Vega_translation
```

Sources starting with Ens_Hs_ are not done by the xref pipeline either.
So warnings:

```
not in the new database but are in the old
```

can be ignored for those.

Some xref are created in the xref_db but never loaded into core, like

```
%_notransfer
```

Warnings:

```
in the new database but NONE in the old
```

can be ignored for those.

Check if the object xrefs are linked to the same object type. If they were moved from Transcript to Gene – this can explain a decrease in object xref numbers.

Object xrefs with ox_status 'DUMP_OUT' will be copied to core. You can compare numbers of object xrefs by ensemble_object_type,
source name and ox_status between the current and previous release by running this query on each xref db:

```
select s.name, ox_status, ensembl_object_type, count(*) from object_xref ox jo
using(xref_id) join source s using(source_id) group by s.name, ox_status, ense
```

Dramatic differences in counts could be explained by a change in the mapping identity threshold applied to sequence mappings for xrefs from a given source.
The threshold is defined in the exonerate method module in XrefMapper/Methods. set_methods in SubmitMapper.pm or species.pm defines which methods are to be used for which sources.

From release 68, a bug to do with how exonerate methods are set was fixed and 100% identity threshold was correctly used for Uniprot xref mappings.

If you're running xrefs on a core db whose xrefs haven't been updated since the bug was fixed, you can use this query to find out the total number of Uniprot object xefs
with target and query sequence mappings < 100% in the previous core db:

```
select count(1), db_name from object_xref join xref using(xref_id) join extern
using(external_db_id) join identity_xref using(object_xref_id) where
db_name like 'Uniprot%' and xref_identity < 100 and ensembl_identity < 100
group by db_name;
```

These numbers should be similar to the decreases you'll see in the warnings.

You can use the xref mind map script (ensemble/misc-scripts/xref_mapping/xref_mindmap) to generate a map of xrefs for a species

and check how xrefs are linked to ensembl genes, transcripts and translations. Xref mind map

If you find a cause for a change in numbers of xrefs from a particular source, this will explain changes in counts for xrefs which are dependent on that source.

NOTE: Xrefs are updated by deleting old xrefs for the sources about to be updated and then adding
the new ones, so if we are not updating a source, it's old xrefs will still stay in the core database.

```
xref_mapper.pl FINISHED NORMALLY
```

If you are happy with the messages we can now transfer the data to the core database. This is done by adding -upload to the command line (see above).

```
Options: -file xref_input -upload
running in verbose mode
current status is tests_finished
```

Report the current status of the xref_database. This is used to work out what to do next.

We can see here that the tests are finished and we are ready to load the data.

```
Deleting data for EMBL from core before updating from new xref database
Deleting data for EntrezGene from core before updating from new xref database
Deleting data for GO from core before updating from new xref database
Deleting data for goslim_goa from core before updating from new xref database
Deleting data for IPI from core before updating from new xref database
```

Delete the data for the sources we are updating.

```
updating (236) EMBL in core (for DEPENDENT xrefs)
DEP 42665 xrefs, 94223 object_xrefs
updating (39) EntrezGene in core (for DEPENDENT xrefs)
```

```
DEP 21473 xrefs, 23897 object_xrefs
        added 30853 synonyms
updating (52) GO in core (for DEPENDENT xrefs)
GO 4535
updating (274) goslim_goa in core (for DEPENDENT xrefs)
DEP 99 xrefs, 96927 object_xrefs
updating (91) IPI in core (for SEQUENCE_MATCH xrefs)
SEQ 35478
```

So we report the number and type of xrefs that are loaded.

```
Building Transcript and Gene display_xrefs using xref database
```

In the official naming routine which mouse, human and zebrafish run, we set the display_xrefs and descriptions.

```
Using xref_off set of 722445
```

So xref_id in the xref database + the offset will be the same as the core xref_id. Used for checking/debuging mainly.

For those that the official naming routine could not set, we now add display_xrefs and descriptions.

```
Precedence for gene display xrefs (1- best name)
        1       RFAM
        2       miRBase
        3       Uniprot_genename
        4       EntrezGene
IGNORE SQL: SELECT DISTINCT ox.object_xref_id
  FROM object_xref ox, dependent_xref dx,
       xref xmas, xref xdep,
       source smas, source sdep
    WHERE ox.xref_id = dx.dependent_xref_id AND
          dx.dependent_xref_id = xdep.xref_id AND
          dx.master_xref_id = xmas.xref_id AND
          xmas.source_id = smas.source_id AND
          xdep.source_id = sdep.source_id AND
          smas.name like "Refseq%predicted" AND
          sdep.name like "EntrezGene" AND
          ox.ox_status = "DUMP_OUT"

IGNORE SQL: SELECT object_xref_id
  FROM object_xref JOIN xref USING(xref_id) JOIN source USING(source_id)
   WHERE ox_status = 'DUMP_OUT' AND label REGEXP '^LOC[[:digit:]]+'

Updated 18855 gene display_xrefs

Precedence for transcript display xrefs (1- best name)
        1       RFAM
        2       miRBase
        3       Uniprot/SWISSPROT
Updated 2918 transcript display_xrefs
```

IGNORE SQL sections list sql statments that select object_xrefs which are not going to be considered while setting display_xrefs.

```
Precedence for gene descriptions (1- best description)
```

```
        1       RFAM
        2       RNAMMER
        3       TRNASCAN_SE
        4       miRBase
        5       HGNC
        6       IMGT/GENE_DB
        7       Uniprot/SWISSPROT
        8       RefSeq_peptide
        9       RefSeq_dna
        10      Uniprot/SPTREMBL
23650 gene descriptions added
```

List of sources which will be used to set gene descriptions.

```
xref_mapper.pl FINISHED NORMALLY
```

The script has finished successfully. If you do not see this then it crashed for some reason and you need to look at the mapper2.err file.

Labels None