



eGenVar Command line tool tutorial

(For versions *beta* 3.00, Last edited 06 August 2014)

Authors: Sabry Razick (sabryr@gmail.com) & Pål Sætrom

Table of Contents

1. Downloading and installing software.....	3
1.1 Upgrading.....	3
1.2. Create an account.....	4
2. Adding file.....	6
1.1. Prepare for adding.....	6
1.1.1. Summary.....	6
1.1.2. Details of commands used to add file details.....	7
1.1.3. Advanced operations.....	8
1.2. Editing description files.	9
1.2.3. Files.....	10
1.3. Adding files.....	11
1.4. Batch operation : adding files recursively.....	13
1.4.1 Current as Batch mode.....	13
1.4.2 Current as Batch mode Current as batch list	13
1.4.2 Current as Batch mode Current as batch list	13
1.5. Undo and rollback.....	13
2. Hierarchical Relationships.....	14
2.1. Create relationship template	15
2.1. Editing the relationship template	15
1.3.1 File relationships.....	15
2.3. Add relationships.....	17
2.4. Rearrange using folder information.....	17
3. Adding data to the database using templates.....	19
3.1. Create template.....	19
3.2. Edit the template	19
3.2. Adding the template content to the database.....	20
4. Tagging with standard information.....	21
4.1. Create new Tag sources.....	21
4.1.1. Template based tag creation	21
4.1. Create tag.....	22

4.1. Add tag to database.....	23
5. Search.....	26
5.1 Search.....	26
5.1.0 Searching for tags.....	26
5.1.1 Example 1: Find out what can be searched.....	27
5.1.2 Example 2: Simple search.....	27
5.1.3 Example 3: Combine query terms.....	28
5.1.4 Example 4: Search using more than one parameter.....	28
5.1.5 Example 5: Search using checksum.....	28
5.1.6 Example 6: Search for distinct matches	28
5.1.10 Example 11: Output the search results as HTML.....	29
5.1.11 Example 12: Get results of a web query using command line. .30	
5,1,12, Search using begins with and ends with instructions to speed up search.....	30
5.2 Batch Search.....	30
5.2.1. Retrieve sample names and write to a text file.....	30
5.2.2. Retrieve file paths of all files connected to any of the sample names listed in a file.....	31
6. Update.....	32
6.1 Auto Update.....	32
6.1.1. Examine the necessary updates	32
6.1.2. Perform the updates.....	32
6.2 Updating a specific table.....	34
7. Delete.....	35
8. Advanced commands.....	36
8.2 Force the output directory*.....	36
8.2 Force the input directory for config files*.....	36
8.3 Force the input directory*.....	36
8.4. Override warnings.....	36
8.5. Check changes.....	36
8.6. Using egenv to check local file duplicates.....	37
8.5. Change query timeout.....	37
9. Error handling and debugging	39
9.1.Working with Write protected directories.....	39
9.2 Errors related to processing large amount of data or long running process.....	39
10. Getting help from UNIX commands when automating.....	42
10.1. Transforming a text file to deed in as search parameter.	42
10.2. Other helpful commands.....	42

The egenv tool described in this tutorial is part of the eGenVar data management system (EGDMS). This tool meant to be used with the EGDMS server and not as a stand-alone tool. When the EGDMS server is started and ready to accept connections a download link to the egenv tool discussed here will be generated and displayed on the server main page.

1. Downloading and installing software

The egenv tool is a command line tool and works on windows and UNIX systems. This tool has local operations and operations that involves communicating with a server. The tool is downloaded from the server the tool will communicate with. Although the egenv tool downloaded from any serve EGDMS will work with other servers, it is recommended to download from the it is bundled with.

1.1. UNIX

- a. Change to the location where you need to install egenv (`cd <location>`)
 - b. Download the package (`wget http://<Server_NAME>:8085/egenv.tar.gz`)
 - c. Uncompress (`tar xvfz egenv.tar.gz`)
 - d. Make the egenv file executable (`chmod +x egenv`)
 - e. Set path (`export PATH=$PATH:./<location>/`)
 - f. Authenticate
 - `egenv -authenticate`
 - The first part of the authentication is to locate the correct server and available methods. For this the *egenv* requires the URL of the WASDL (Web Services Description Language) file. This URL is generated after the server is installed and will be displayed in the web interface index page. This URL will also be sent via email to the
- ### 1.7. Windows <TODO>
- a. . Download the package to the installation location
 - b.. Uncompress the content using a ZIP utility
 - c. Authenticate against the server as described for the UNIX version of the tool

1.1 Upgrading

- `egenv -v`

- This will print the current version and also prompt for an upgrade if there is a newer version available.
- **egenv -upgrade**
 - Upgrade to latest version, if a newer version is available
- **egenv -upgrade -force**
 - Replace with the current version available from the server, regardless of the local version.

1.2. Create an account.

There are three ways to create an account. First, use the administrator menu if you have access to it or request an account from someone who has access (explained in the administrators manual). Second, use the web interface as described in the web interface tutorial. The, use the egenv command line tool. To use the egenv tool, you should know the location of the WSDL. The URL for the WSDL is generated during installation and emailed to the administrator. Usually the format of the URL is

http://<Server_NAME>:8085/eGenVar_WebService/Authenticate_service?wsdl

Steps:

1. **egenv -request_account your_email@mail.com**

.Waiting for the server ..

...._____

**Please enter the verification code (sent to the email
sabryr@gmail.com)
(press C to cancel or provide email):**

Issue the -request_account instruction with a valid email address. Then an email with a verification code will be sent to this email. When you receive the code enter it in the space provided.

If the code was correct you will see the following message

.Waiting for the server ..

**.. An account was created and login details mailed to
sabryr@gmail.com**

2. Adding file.

The files are organized into reports and reports are grouped in to report batches.



Adding files will require the following changes to the database

- Create or select a Report batch.
- Create or select a report.
- Add details about the files to the database. All these steps can be performed with a three-step operation with the egenv tool.
 - a. Prepare: Create the description files. These describe the above mentioned Report_batch, Reports, File type and the files.
 - b. Edit: The description files could be edited to add custom details (optional).
 - c. Add: Add the details to the database.

1.1. Prepare for adding

This step will create a description file to be used by the add step. The description file will contain details about the report batch, report, sample details, and files.

1.1.1. Summary

The following is the list of commands to issue. For details refer to section 1.1.2.

- Change the current directory to the directory where the files are.
- **sh egenv -authenticate**
- **sh egenv -prepare**
 - Name selected for the Batch: **Y** (accept the default name, the name of the parent directory will be the default value)
 - Name selected for the Report.....: **Y** (accept the default name, the name of the current directory will be the default value)
 - There are subdirectories under ...: **no-all** (do not use content of sub-folder, relevant only when there are sub-folders)

Two files will be created at the end of this steps. (/.../batch_1/report_1/ is the current directory where the egenv was executed from)

- /.../batch_1/report_1/egenv/.egenvar_describe_files_warnings.txt
- /.../batch_1/report_1/egenv/.egenvar_describe_files.txt

1.1.2. Details of commands used to add file details.

- Change to the directory where the files are. The commands must be executed from the location where the actual files are located. If links (linked file) are found, the actual files would be resolved before proceeding.
- **egenv -authenticate**
 - (To make sure the user and location are authenticated). This makes it easier to debug errors in the later steps. If the authentication fails none of the remaining steps will work. Stop here and contact the administrator.
- **egenv -prepare**
 - This prepares the current directory and its contents for adding to the database. When this command is issued, first the user will be warned about the possibility of losing results from earlier preparation for this location.
 - Recreate mode. All changes to
 - ../.egenvar_describe_report.txt,
 - ../.egenvar_describe_report_batch.txt files
 - .egenvar_describe_files.txt,
 - .egenvar_describe_filetype_description.txt
 will be lost. Press, Y to confirm|N to cancel :
 - Press the **Y** and press **enter** to allow a clean prepare.

- Name selected for the Batch was rand. Y-to accept this|N -cancel| or enter a new name:
 - Here the user must select a name for the batch. The suggested name is the name of the parent folder. If the default name is selected by pressing Y, then the parent folder name will be used as the batch name. Using the parent folder name is highly recommended as this allow the software to take automatic actions in later stages. If the user wants to change the name, then that name could be typed here. If the user wants to append to an existing batch, then the exact name of that batch should be typed in here. If a name is typed instead of selecting the default by pressing Y, then the egenv will suggest that the parent folder name to should be changed accordingly. If the user selects Y, then the parent folder would be renamed. When a name is typed it is always checked against the database and the user will be notified about the findings. To make things simple we will select Y and use the parent folder name as the batch name.
- Name selected for the Report was egenv: Y-to accept this|N -cancel| or enter a new name:
 - Here a name for the report is requested and the same rules as for the batch name selection applies here as well. We will press Y and select the current folder name as the report name.
- There are subdirectories under the folder './../rand/egenv/lib', do you want to use all files in all subdirectories ? Y (yes)| N (no)| all (use_all_subdirectories| no-all (skipp_all_subdirectories)| :
 - This message will appear only if there are subfolders in the current directory. If the user wants to include the sub directories as well, then that can be done here. If the there are more sub directories inside the subdirectories the user will be prompted each time. To avoid this prompting the user can select **all** or **no-all** options. The **all** would consider all subdirectories while the **no-all** would avoid all subdirectories.
- Now there should be 2 files created with descriptions required in the add step.

1.1.3. Advanced operations.

- Overriding the input directory.
 - The default input directory (the directory to be processed) is the current working directory (obtained with the *pwd* command). To change this use the *in* instruction
 - **egenv -prepare -in /home/files/batch_7/**

- The above command will process the /home/files/batch_7/ directory instead of the current one.
- Override where to write the config (prepare files) files to.
 - The default directories to write the output files is the current working directory and its parent directory. To change this use the *configout* flag.
 - **egenv -prepare -configout /home/files/batch_7/config/**
 - The above command will write all outputs to the /home/files/batch_7/config/ directory instead of to the default ones.
 - **egenv -prepare -clean -out /home/files/batch_7/config/ -in /home/files/batch_7/**
 - The above example shows how to change the default input and output in the prepare step.
- Use default values without confirming first. This will use the current folder name as report name and the parent folder name as the report batch name. This will also delete any previous configuration files created. **Note: use with caution.**
 - **egenv -prepare -default**
 - **egenv -prepare -default; egenv -add**
 - This will automatically add all content from the current directory with default values without confirming anything from the user.

1.2. Editing description files.

The description files have three types of information lines. Comments starting with the "#", column titles starting with "##", and data lines that do not start with a "#". Each column is separated by a tab and each line is ended with the new line. Only UTF-8 plain text allowed.

- The fields (columns) marked with **** are identifiers assigned by the database. Positive value indicates that the entry already there in the database and none of the values in the same row should be edited. Only the rows which has a negative value for this field should be edited. All edits to rows with a positive value for this field will be ignored. **However, for intentional change of associations values in the same row could be edited, i.e. attach a file to multiple reports** (refer section 1.2.3 for more details).
- The fields with * are compulsory and can not be left blank.
- The fields with ** are compulsory and value are pre-selected. These values should not be modified.
- The fields with *** are compulsory and refer to existing entries in the database. when changing this values you have to make sure they exist in the database (use search option on egenv or the web interface), otherwise the whole operation will fail.
- The fields which are not marked are free to edit. A maximum of

1.2.3. Files

file :.egenvar_describe_files.txt

This is the file contains file metadata including the path information, sample information, report name, report batch name and the email of the person reporting the information. The only obligatory fields required for the reporting are the file checksum value and the full file path.

However, it is highly recommended to include the files in a report.

(**Note:** in earlier version this information was provided in three separate files, but changed to be in one file as this proved more convenient for people generating this file from their own scripts or programs)

- **FILES.ID******: This is the file id if it already there in the database. The file is fetched using the checksum value. Therefore, if a file in some other location has the same checksum, it could also appear here with its original report name. It is highly recommended have only one copy of a file. But for some practical reason multiple copies needs to be maintained then this could also be facilitated with the system.
- **FILES2PATH.LASTMODIFIED**: Extracted from file metadata and should not be modified
- **FILES.CHECKSUM*** : This is a SHA1 checksum value in 64 bit format, excluding the padding characters. Calculated and should not be modified. This is the key in the files table (given checksum can appear only once). If there are multiple copies of the same file exists in different locations, the there will be one entry in the files table and multiple entries in the files2path table corresponding to that entry.
- **FILES.FILESIZE**: Files size in bytes. Extracted from file metadata and should not be modified
- **FILES2PATH.OWNERGROUP**: The group which has read access to this file. Extracted from file metadata and should not be modified
- **FILES.NAME** : The absolute path of the file as provided by the file system. All symbolic links will be resolved to the original location.
- **REPORT_BATCH.NAME***** : The name of the report batch.
 - If this name is new then a new report batch will be created.
 - If this name already found, then the information will be appended.
- **REPORT.NAME***: the name of the report. the name should be described in
 - If this name is new then a new report will be created.
 - If this name already found, then the information will be appended.

- If the file is already found to be attached to a different report in the database, then a new relationship links will be made. i.e. the file will be part of both reports. This is useful when data from multiple experiments are found in one file.
- **FILES2PATH.LOCATION***: The server name obtained from the operating system. A effort will be taken to obtain the full address of the server that can be referenced from the network.
- **PERSON.EMAIL**: The email address of the person reporting the file. If there is no profile yet for this email a temporary one will be created (this email will not be associated with a login account).
- **FILES2PATH.FILEPATH*** : Same as **FILES.NAME**.

Optional fields. This columns could be excluded if not relevant.

- **HOSTNAME** : Same as FILES2PATH.LOCATION, included for backward compatibility
- **SAMPLEDETAILS.NAME**: The name of the sample details entry in the database. If there is no entry yet a temporary one will be created. Use the text NA for this field if this information is not relevant or not available.
- **REPORT.ENTRYDATE**: The date this entry is created. This a generated value using system data and time.

1.3. Adding files.

After the prepare step, the .egenvar_describe_* files could be edited to include any information missed during the preparation step. After that issue the following command to add the files, from the same directory where prepare was performed.

- **egenv -add**

The command will add all files in the current directory and recursively in the sub-directories which are included in the .egenvar_describe_files.txt file. If only some of the files in the directory are to be included in the current adding operation then those ones should not be added could be removed from the .egenvar_describe_files.txt or commented out (adding a # at the beginning of the line). Alternatively, wildcard characters could be used as follow.

- **egenv -add *exp1***
 - (add all files that has the term exp1 in their file name)
- **egenv -add experiment_?.fa**

- (add files with the term experiment_ and ends with .fa and has any single character in the place of the ? sign. e.g. experiment_1.fa,experiment_A.fa,experiment.fa)

At the end of the adding operation the following could be the result:

- The details about the files added successfully as new entries.
- The files were already found
- **egenv -add -configin /home/new/files**
 - This command will add content of the /home/new/files/ directory instead of the current directory.

1.4. Batch operation : adding files recursively

This facility allows to add files on many folders at once. This could also be used to add the content of an entire storage at once as well. The best way to add many folders at once is to use symbolic links. Create a directory with write permission and then place symbolic links to target folders in this location. Then use one of the options given below to create the template. The **-recurse** flag is compulsory for this operation and additional **-split_limit** and **-interval** flags can be combined for finer control. After preparing the adding step is same as for the normal mode (just use **egenv -add** from the directory where the command was executed)

1.4.1 Current as Batch mode

In this mode a report batch will be created with the same name as the current folder, then one report each for each subfolder. All files in each subfolders will be added to report corresponding to that subfolder.

- **egenv -prepare -recurse current_as_batch**

1.4.2 Current as Batch mode Current as batch list

This mode is well suited when the current directory contains symbolic links to actual locations to be processed. One report batch will be created for each subfolder in the current folder and reports will be created each sub-sub folder.

- **egenv -prepare -recurse current_as_batch_list**

1.4.2 Current as Batch mode Current as batch list

When there are very large number of files, it is always a good idea to perform a check before creating hashes for all files (which may take many hours). The following command lets you perform a preliminary analysis without calculating the checksums. This will help to identify possible read permission issues etc..

- **egenv -prepare -recurse -nochecksum current_as_batch_list**

or

- **egenv -prepare -recurse -nochecksum current_as_batch**

1.5. Undo and rollback

If there were errors or situation where a solution could not be found without getting more information from the user, then a rollback is called. All changes to the database are removed. If the user wants to undo after the changes were committed then an undo operation is provided. Every Time an add operation is performed a undo point is created

with the file name format .<date_time>_egenvundo.txt. The undo operation can be called with this file as input. e.g to call with the file “.2013_03_27_12_35_55_egenvundo.txt”

- **egenv undo .2013_03_27_12_35_55_egenvundo.txt**

When this command is issued, all newly created entries which are described in the undo files will be removed from the database if it safe to do so. If there are subsequent changes to the database makes it unsafe to perform this operation then none of the records will be deleted. Therefore this is a all or nothing operation. If there was something wrong, you have to undo this immediately before anyone else makes any changes that might be affected.

2. Hierarchical Relationships

Hierarchical relationships could be added to following record types

- files
 - When a file(parent) is used to create another file (child) hierarchical relationships can be constructed to represent this information. For instance when a software takes 2 files as input to generate a result file. The two input files are the parents of of the result file.
- report
 - When the data in a report (child) is generate using or extended from another report (child). This could be used to constructed relationships between two groups of files as well.
- report_batch
 - The relationship at report batch level is provided to complete the set of relationships but no use of it is defined.
- sampldetails
 - When a sample(parent) is divided to make sub-sample or samples are combined to make a single sample, this relationship can be used.
- donordetails
 - This relationship takes the form organism -> organ -> tissue ...and provides way of grouping samples from same tissue or organ etc..

The construction of relationships has three steps.

- Create template

- Edit the template
- Add to database

2.1. Create relationship template

- **egenv -prepare -template**

This command will list tables to be used for the template . Alternatively the table could be specified as in the examples below.

- Files (files_hierarchy). This operation should be performed from a directory with write permission.
 - **egenv -prepare -template files_hierarchy**
- reports (report_hierarchy)
 - **egenv -prepare -template report_hierarchy**
- report_batch (report_batch_hierarchy)
 - **egenv -prepare -template report_batch_hierarchy**
- sampledetails (sampledetails_hierarchy)
 - **egenv -prepare -template sampledetails_hierarchy**
- donordetails(donordetails_hierarchy)
 - **egenv -prepare -template donordetails_hierarchy**

2.1. Editing the relationship template

There are two columns in the template files. First one is the Parent_name and the second is the Child_name. The names should be exactly as it appears in the database. Up to 999 rows of relationship information can be included in a single file. Each column is separated by a tab. The topmost comment line indicates the type of relationship and this should not be edited. e.g. "#relationship_type=files_hierarchy".

1.3.1 File relationships

This example show how to translate the relationships show in the diagram below to a hierarchical relationship in the EGDBMS.



The template files_hierarchy file for the above relationships is shown below. The names used to refer to the files is the value found under the name field of the file. Similarly, for reports, report_batches and other entities where hierarchical relationships the exact value found under the name field must be used.

#relationship_type=files_hierarchy	
#parent child	
SCP:server1.edu:/home/user1/raw_data.txt	SCP:server1.edu:/home/user1/rgrromed.txt
SCP:server1.edu:/home/user1/grromed.txt	SCP:server1.edu:/home/user1/precessed.txt
SCP:server1.edu:/home/user1/precessed.txt	SCP:server1.edu:/home/user1/finterpret.txt

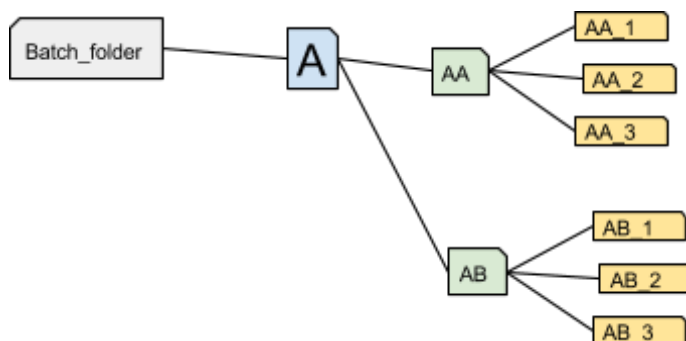
2.3. Add relationships

Adding operation is the same for all hierarchy types. Call the command with the file with the relationships.

- `egenv -add -template .egenvar_describe_relate_files_hierarchy.txt`
- `egenv -add -template .egenvar_describe_relate_report_hierarchy.txt`
- `egenv -add -template .egenvar_describe_relate_report_batch_hierarchy.txt`
- `egenv -add -template .egenvar_describe_relate_sampledetails_hierarchy.txt`
- `egenv -add -template .egenvar_describe_relate_donordetails_hierarchy.txt`

2.4. Rearrange using folder information

The eGenVar system is designed to extract information from current organizational practices. The function described below uses folder arrangement to create hierarchical relationships between the reports. For example lets use the directory structure given in the figure below.



The figure show a arrangement of subfolders. Each folder has files(not shown) in addition to the folders shown. The task is to reflect all this information in the EGDMS with minimum effort.

The first step is to create the report batch and one report . Then add all the files in the subfolders recursively.

- Change to the “Batch_Folder” directory and execute the prepare command
- `egenv.sh -prepare`
 - Prepares for adding using **Batch_folder** as batch name and **A** as report name.
- `egenv.sh -add`
 - Add the information in the templates generated during the above step to the database.

- At the end of the above two steps a report named **A** will be created under the report batch named **Batch_folder**. All the files in all subdirectories of the folder A will be added to the report named A.
- Now from the directory **Batch_folder**, issue the command below to re-group the content.
- **egenv.sh -prepare -hierarchy -depth 2**
 - This command prepares a template with rearrangement information. -hierarchy instruction arranges the subfolders as child reports considering the parent folder as the parent report. If the child report does not exist in the database they will be created. Parent reports will not be created unless they are a direct child of an already created report.
 - The -depth instruction limits the number of levels the algorithm uses. In other words, how deep into the subfolder hierarchy the rearrangement takes place is decided by this.
- **egenv.sh -add -template .egenvar_report_relate.txt**
 - This step adds the information in the template to the database. The .egenvar_report_relate.txt is the template file generated in the above step.
 - At the end of this step two new reports AA and AB will be created as children of A. Then AA_1, AA_2 and AA_3 child reports will be created using AA as parent report. Same procedure will be followed for AB as well.
 - After the report hierarchy is established, all the files which were initially under the report A will be re-grouped under appropriate reports using the file path information.

3. Adding data to the database using templates.

This section describes how to add more details to the database and then to construct relevant connections. The tables could be populated using a template based approach as described in earlier sections. For example lets see how to add data to the sampledetails table and

3.1. Create template

- **egenv -template**

Select the table sampledetails from the table list to create a template for the sample details.

- **egenv -template sampledetails**

The file .egenvar_describe_sampledetails.txt will be created.

3.2. Edit the template

```
#The fields with **** are identifiers assigned by the database. Positive value indicates
# that the entry already there in the database and none of the values in the same row
# should be edited. Only rows which has a negative value for this field should be edited.
# However, for intentional change of associations values in the same row could be edited,
# i.e. attach a file to multiple reports
#The fields with * are compulsory and can not be left blank.
#The fields with ** are compulsory and value are pre-selected. These values should
# not be modified
#The fields with *** are compulsory and refer to existing entries in the database.
# Please make sure the entry exist in the database. If a value entered here was not found,
# an attempt would be made to first create an entry with the value.
#For non compulsory fields ,use NA when you do not have value.
#
##sampledetails.name* sampledetails.description
unique_name_002 This sample was obtained from X and ...
```

In the above example all content starting with the # are generated when creating the template. The last line (in red) needs to be added by the user. Please note as do all other files generated by the system, the columns are separated by a single tab character (press the TAB key once)

3.1. Create template using search

- `egenv -template -search report.name=test`

The above command will create a file “.egenvar_describe_report.txt” with reports matching the search criteria. This file created can be used for deleting or updating files. this method is useful when large number of records needs to be updates at once.

3.2. Adding the template content to the database

- `egenv -add -template <template filename>`

For the sample details this would be

`egenv -add -template .egenvar_describe_sampledetails.txt`

4. Tagging with standard information

tagging is performed for files,reports,report batches, sample details, donors,persons and other entities defined as taggable. This process allow the data inserted to the database to be connected to publicly available data with minimum effort. Open ontology, minimum information for reporting , disease ontology and some standard tag sources are provided with the system. In addition the administrators can introduce new tags as appropriate.

4.1. Create new Tag sources

Two methods to introduce new tag sources . First method is to create a files with the tags and upload it when starting the server and described in the administrator's manual. The second method is to use the command:

- **egenv -create_tag_source <TAG_SOURCE_NAME>**

This command will initiate a new tag source. After the initiated tags can be created using a template based approach or the source can be populated when a new tag is used for the first time. his operation is allowed only to users with level one or higher clearance (level 0 & 1). This is to restrict the creation of redundant tags and to enforce consistence.

4.1.1. Template based tag creation

This example show how to add tag s to the tag source. The file should be simple text (if created using a spreadsheet program or word processor, should be saved as simple text). There could be as many number of rows, but it is recommended to split the file if you have more than 1,000,000 rows.

```
##SAMPLEDETAILS2TAGS.LINK_TO_FEATURE
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Microarray_Scanner_System/Illumina/HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Microarray_Scanner_System/Illumina/iScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/sequencing/sequencer/Illumina/MiSeq2500
```

In this example the file is saved with the name add_instrument.txt and the following command is issued to update the tag source in the database. Duplicates are automatically avoided so issuing the command multiple times on the same file will not cause errors. At the same time if you want to change a tag you should delete the old one first (there is no update feature at the moment) .

- **egenv -template -add /home/sabryr/tmp/hunt/tags/add_instrument.txt**

4.1. Create tag

egenv -prepare -tag

The example below show how to tag a report containing sample annotation of a microarray experiment.

What type of tag do you want to create?

0) ADDRESS_TAGSOURCE

2) FILETYPE_TAGSOURCE

4) HUNT_PROJECT_ID_TAGSOURCE

6) IDENTIFIERS_TAGSOURCE

8) MIBBI_TAGSOURCE

10) PROJEC_IDENTIFIERS_TAGSOURCE

Q) Quit

1) BIOLOGICAL_ONTOLOGIES_TAGSOURCE

3) HUMAN_DISEASE_NTOLOGY_TAGSOURCE

5) HUNT_TAGSOURCE

7) INSTRUMENTS_TAGSOURCE

9) MICROARRAY_TAGSOURCE

11) REPORTTYPE_TAGSOURCE

Selection: : 8

What type of tag do you want to create?

0) MIFlowCyt

2) MIMix

4) MIBBI

6) CIMR

8) MINSEQ

10) MIASE

X) None of the above or more than one possibilities, use the parent value(MIBBI_TAGSOURCE=0)

Q) Quit

1) MIAME

3) MIARE

5) MIABE

7) MIGS-MIMS

9) MINEMO

Selection: : 1

What type of tag do you want to create?

0) The experiment design including sample data relationships

1) The final processed data for the set of hybridisations in the ex

2) The essential sample annotation, including experimental factors

3) The raw data for each hybridisation

4) Sufficient annotation of the array design

5) Essential experimental and data processing protocols

X) None of the above or more than one possibilities, use the parent value(MIBBI_TAGSOURCE=721)

Q) Quit

Selection: : 0

Please select the target table (e.g. when tagging files select files2tags)

0) FILES

2) REPORT_BATCH

Q) Quit

1) REPORT

3) SAMPLEDETAILS

Selection: : 1

How do you want to select the REPORT2tags to be tagged ?

0) Perform a search on REPORT

1) Use a SUC (static URL code) to select the REPORT to be tagged

2) Do not search. (Add the REPORT values manually)

3) Get me out of here, do not want to make nay changes

Selection: : 0

Enter the name/names of the REPORT

Search for: : experimental_design_info

```

Do you want to create one entry in the REPORT2tags table or create a template to enter many entries?
Y-create one entry | T-create template file | C- cancel and quit: t
Creating template started @2014-02-03 17:52:08
..

.egenvar_describe_REPORT2tags.txt -- created
Ended in 7 minutes :24 seconds

```

4.1. Add tag to database

egenv -add -template <tag_file>

for the above example this would be

egenv -add -template .egenvar_describe_report2tags.txt

4.1. Batch tagging

The following example show how to attach information to sample details using tags. The example files are from a microarray experiment. The information are extracted from sample sheets generated from the Genome Studio software (Illumina), and translated in to tag sources, before attaching as tags.

The header of the sample sheet is shown in the box below.

```

[Header],,,,,,,,,,
Investigator Name,SEE,,,,,,,,,
Project Name,2013_10033_HUNT_CVD,,,,,,,,,
Experiment Name,2013_10033_HUNT_CVD,,,,,,,,,
Date,20131202,,,,,,,,,
[Manifests],,,,,,,,,,
A,humancoreexome-12v1-0_c,,,,,,,,,
[Data],,,,,,,,,,

Sample_ID,Sample_Plate,Sample_Name,Project,AMP_Plate,Sample_Well,SentrixBarcode_A,SentrixPosition_A,Scanner,
Date_Scan,Replicate,Consentration,Parent1,Parent2,Gender

```

First in this example we will attache instrument related information to the sample, specifically the scanner used. For this the columns `Sample_ID` and the `Scanner` columns needs to be extracted. The `sample_id` will be used to identify the correct sample in the database.

When there is no instrument related information yet in the EGDMS the tag source needs to be created by a user with administrator privileges. The rest of the example assumes that the “create new tag source” was successfully performed, if it failed the rest of the example will not work.

The sample sheet will only give the names of the scanners, the tags how ever, needs to more descriptive. For example, if the scanner name is “HiScan”, then you will need at least find out which company makes it and whether there is some order in the design name assignment. In the box below, a simple arrangement for this shown with the possibility to expand in the future. For instance, if a different scanner from the sample company or a scanner from a different company is to be used, then this tag source created should be expandable without creating a new one. The point of tagging without including this information as text in the description for sample is, reusability to maintain connections and to expand in the future.

```
##SAMPLEDETAILS.NAME SAMPLEDETAILS2TAGS.NAME SAMPLEDETAILS2TAGS.LINK_TO_FEATURE*
CVD_1045860123383 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Microarray_Scanner_System/Illumina/HiScan
CVD_1045860074326 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Microarray_Scanner_System/Illumina/HiScan
CVD_1045860092955 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Microarray_Scanner_System/Illumina/HiScan
CVD_1045860052034 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Microarray_Scanner_System/Illumina/HiScan
CVD_1045860043056 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Microarray_Scanner_System/Illumina/HiScan
```

the INSTRUMENTS_TAGSOURCE is the tag source name created by the administrator. This tag source at the moment may contain only the super parent label “INSTRUMENTS” which will have the id=0 and also the parent_id=0. This will be automatically added to the tag source when it is created. the label “Microarray_Scanner_System” is used next to provide room for other types of instruments categories to be added later. The the label “Illumina” provides manufacturer. After the scanner name is provided. another possible order for this is given below.

```
##SAMPLEDETAILS.NAME SAMPLEDETAILS2TAGS.NAME SAMPLEDETAILS2TAGS.LINK_TO_FEATURE*
CVD_1045860123383 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Illumina/Microarray_Scanner_System/HiScan
CVD_1045860074326 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Illumina/Microarray_Scanner_System/HiScan
CVD_1045860092955 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Illumina/Microarray_Scanner_System/HiScan
CVD_1045860052034 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Illumina/Microarray_Scanner_System/HiScan
CVD_1045860043056 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Illumina/Microarray_Scanner_System/HiScan
CVD_1045860107604 HiScan
INSTRUMENTS_TAGSOURCE/INSTRUMENTS/Illumina/Microarray_Scanner_System/HiScan
```

Add this to the database,use;

egenv -add -template <tag_file>

1. Set the parent_id of HiScan to Microarray_Scanner_System id (currently the parent_id is that of Illumina)
2. Set the parent_id of Microarray_Scanner_System to Illumina id (currently the parent_id is that of Microarray_Scanner_System)
3. Set the parent_id of Illumina to INSTRUMENTS id (currently it is Microarray_Scanner_System).

At the end of the above three operations (which could be performed using -update instruction in the egenv tool or the update page of the GUI). none of the id of the tags would have been changed only the relationships will. So even if the tags are being used currently, there would be no effect on them.

Warning !. In the above simple example, the change in the relationships was straight forward. however, when changing more complex relationships, for example to reflect a update in the biological ontologies, the children and parents of all nodes to be changed should be carefully examined making sure there are no unintended relationship changes.

Caution ! : When a new child is being attached to a parent, if there is already a child with the same name, this will not cause any issue when creating the relationship. However, when user selecting the tags he/she will see two entries with the same name, and can cause ambiguity. Although this ambiguity can be resolved by inspecting the children and the parents, creating such relationships should be avoided.

In this example the manufacturer is given write after the instrument label, to arrange the grouping according to manufacturers. The arrangement best suits can be adapted. In the future if it proves to difficult add a new tag or a better arrangement is suggested, then the hierarchy can be rearranged without affecting the used tags. This is an advantage of this type of tagging. For instance, you could easily go back and forth between the two arrangements provided without affecting currently used tags.

For example to change from order one to the second you need, recreate the parents before changing the references.

5. Search

5.1 Search.

The search takes the form .

- **egenv -search <source> <target>**

The only difficult part in the search operation may be to find out what are the names the real world objects are known in the database. To overcome this, the natural name is used whenever possible. For instance files are called *files* and a report is called *report*. However when you want to search sample information you should use *sampledetails* (as the name sample is already taken). These names can be found out using the *-available* operation and explained below.

5.1.0 Searching for tags

The search for files using tags could be done using the following commands

- **egenv -search TAG=cancer files**

To search through all tagable entities e.g. files sampledetails, reports,... (can be slow if the database is big!)

- **egenv -search TAG=cancer**

If searching for Reports, Report_bacthes, Sampledetail or any other tagable entity then the entity name must be given. The default (as shown above) when no target is mentioned is to search through all tagable entities. To restrict this for reports

- **egenv -search TAG=cancer report**

The search can be made more specific by specifying the parent. For example the example below search for a Sentrix barcode position (as identification method of the locations of microarrays). The position in this search is "R01C01" and the sentrix barcode is the "4515597045". i.e. The position "R01C01" is a child of "4515597045" this tag source.

- **egenv -search TAG=451559704.R01C01**

To get all samples with this tag

- **egenv -search TAG=451559704.R01C01 sampledetails**

*See batch operations (section 5.2.) to see how to feed in a file with many values as the search parameter.

*The expansion query format is slightly different for tag based queries than for other ones. The following example shows how to expand the query to get files using a sampledetails tag. i.e. find all sampledetails which is tagged with the given tag, then get all files connected to those samples.

- **egenv.sh -exact -search tag=7616836088.R01C01 "SAMPLEDETAILS|FILES"**
 - The term "SAMPLEDETAILS|FILES" implies that the sampledetails should be found first using the tag given, before looking for the files. Take note that the files are not tagged with R01CO1 in this example and looking for files using this tag will not return anything.

Alternatively if to search through all tagble entities and find files afterwards, the following query can be used. This could take a lot more time to finish as it will search through all tagged entities and will try to locate the files using the graph

- **egenv.sh -exact -search tag=7616836088.R01C01 "|FILES"**
 - This is very slow and would consume a lot more resources and should try to avoid

5.1.1 Example 1: Find out what can be searched

Find out if a file named control_yaleb_chr1.bed exists in the database then the following steps could be followed .

- Find out the field to search for.
 - **egenv -search -available**
 - Locate the correct format from the list displayed.
 - "FILES.NAME__The name for the file. This is usually th.."

5.1.2 Example 2: Simple search

Get all files under the report report_51.

- Find out the field to search for.
 - **egenv -search -available**
 - Locate the correct format from the list displayed
 - "report.name__Report name".
 - "files.name____file name or the location path".
- Search 1
 - **egenv -search report.name=report_51 files**
 - This will give all details of files grouped under the report_51. If you want only the file names then.
 - **egenv -search report.name=report_51 files.name**
- Search 2
 - **egenv -search files.name=control_yaleb_chr1.bed**
 - Note that the target is blank in the query, indicating that source and the target are the same. Alternatively, the following queries will also return the same result

- **egenv -search files.name=control_yaleb_chr1.bed files**

5.1.3 Example 3: Combine query terms

There are two ways to search for multiple things at the same time. The first is to use the “||” (two pipes) to separate the queries and then include all the queries within quate.

- **egenv -search files.name="genrat_404.txt||genrat_479.txt||genrat_607.txt"**

The second is to use wild cards.

- **egenv -search files.name=genrat_***
- **egenv -search files.name=genrat_???.?**

5.1.4 Example 4: Search using more than one parameter

Search for files stored in the server “dmed4929” with the folder name “rand”.

- **egenv -search "files.name=dmed4929&&files.location=rand"**
 - Note the whole <source> is within quotes not just the query.

5.1.5 Example 5: Search using checksum

Find the file with the checksum 82a5e7f499077fbe6dbeeed1543bb2d8

- **egenv -search -exact files.checksum=82a5e7f499077fbe6dbeeed1543bb2d8**
 - Note the -exact flag to perform an exact match search.

5.1.6 Example 6: Search for distinct matches

To find out the distinct set of report type ids used.

- **egenv -search report.name=rposrt_1 report.distinct reporttype_id**

5.1.7 Example 7: Search for minimum or maximum values

To get the minimum value for id column in the report table

- **egenv -search report.id.min**

To get the maximum value for id column in the report table

- **egenv -search report.id.max**

5.1.8 Example 8: Queries with spaces

Replace all spaces with a “\s”. Enclose the query in quotes.

- **egenv -search HUNT_TAGSOURCE.name="Breast\sFeed\sChild"**
HUNT_TAGSOURCE.name

Alternatively use the “%” character, this will match any character not only spaces. (quotations are optional)

- **egenv -search HUNT_TAGSOURCE.name=Breast%Feed%Child**
HUNT_TAGSOURCE.name

5.1.8 Example 9: Numerical search fields

When a numerical (integer, double) column is searched special rules apply. The user does not need to distinguish the search in any way, rather the rules are triggered by system.

Rule 1. Only exact matches are allowed (even when the -exact instruction is omitted).

Rule 2. in addition to ‘=’ numerical matches such as <, > <= are also allowed

- **egenv -search "files.size<1350" files.name**
 - **Note that the usage of quotes**

5.1.9 Example 10: Expand to get more details of foreign keys

When the query **egenv -search report.name=myreport** is used the results will have foreign key references (e.g. REPORT_BATCH_ID,REPORTER_ID) . If the details for this IDs are needed the **-expand** instruction could be used. For example the email address corresponding to the REPORTER_ID could be retrieved this way. Please note that the queries using -expand are slower in comparison. The instruction should never be used when creating templates for deleting records, as this may have unintended consequences.

- **egenv -expand -search report.name=<report_name>**

5.1.10 Example 11: Output the search results as HTML

When the output file specified using the -out instruction and the filename has the .html extension, the out will be in html format (can view using a web browser). Optionally you could specify the instruction -html, if your output file can not be given the extension .html, for some reason.

- **egenv -search report.name=rposrt_1 report -out report.html**
- **egenv -search report.name=rposrt_1 report -html -out report.web**

5.1.11 Example 12: Get results of a web query using command line

This feature allows to locate records using the web interface and then retrieve them using the command line tool. Some use cases for this are;

- A. A user locates the records using the web interface and use the quick reference code to request the records from the administrator.
- B. An outside user (who do not have an account) finds that there are records of interest in the server and requests more details from the administrator.
- C. A user finds out that his/her query returns too many results to be evaluated using the web interface and thus wanted to use the command line to perform batch operations on the results.

The method to do this is to perform a search using the hash generated during the web search interface.

- **`egenv -search code=9ea82d3ca8e7a3868d5fb5f83ad07692819cd8fc`**

To expand the results from a public search requires two steps:

- **`egenv -dontgroup -nolable -search code=9e66324eba7a50617232f8f88ae8279f3f60c1b8 TAGS2HOST.NAME -out tmp.out`**
 - This extracts the public tag names to a file
- **`egenv.sh -dontgroup -nolable -file -search tag=tmp.out "sampledetails|files.name" -out tmp.out2`**
 - This uses the file created in the previous step as input , find all the samples annotated with the tags and writes a relevant files attached to those samples to a file,

5.1.12, Search using begins with and ends with instructions to speed up search

By limiting the wildcard usage by the system, it sometime possible to improve query performance. For example when searching for all duplicate paths of file named "file1.txt" the following query could be used.

- **`egenv.sh -search files.name="file1.txt" files2paths.filespath`**
 - The system will perform the search "%file1.txt%", to look for file1.txt in anywhere of the file name
- **`egenv.sh -search files.name="file1.txt$" files2paths.filespath`**
 - The system will perform the search "%file1.txt", to look for file ending with file1.txt

5.2 Batch Search.

The batch search is specially useful when the egenv is used as a part of of a workflow or when used as a metadata resource to fetch the original data. In this mode either or both input and output are directed to a file.

5.2.1. Retrieve sample names and write to a text file.

In this example all sample names with the term “exp2345” in their name are retrieved and written to a file called samples.txt.

- **egenv -nolables -search sampledetails.name=exp2345 sampledetails.name -out samples.txt**
 - **-nolables** : omit all the column headers
 - **-out sample.txt** : -out directs the out to a files, the immediate argument following the -out is the name of the output file

5.2.2. Retrieve file paths of all files connected to any of the sample names listed in a file.

In this example, a file with list of sample names are provided as input (the file created in the above example, 5.1.9). If you are preparing this file manually of from another program, make sure it is a simple text file and each search term is terminated by a new line(see the example in box below). When using content from a file to search, only an exact match is performed.

```
#sample details names, this is an optional header
exp2341
exp2342
exp2343
exp234AAS
combined_exp234
```

- **egenv -nolables -file -search sampledetails.name=sample.txt files2paths.filepath -out filepaths.txt**
 - **-nolables** : omit all the column headers
 - **-file** : retrieve search terms from a file. i.e instead of searching for the term “sample.txt”, open the file named “sample.txt” and use its content as search terms.
 - **-out filepaths.txt** : -out directs the out to a files, the immediate argument following the -out is the name of the output file

6. Update

As do many other operations the updates are also done where a template-edit-execute series.

6.1 Auto Update

The updates found using the *diff* operation can be updated automatically. This update can be used as a synchronization method. During the diff and update some special terms are used and explained below.

- Path : The real file path (provides the location of the files).
- Host : The address or name of the server hosting the files.
- Location : Host + Path. If two copies of a file exists in two different hosts, there will be two different filepath entries in the files2path table.

Following instructions are available for finer control of the operations

- **-recurse** :- Use all files and folders recursively. i.e. if there a folder in the current path visit that directory and collect all files in it and so on.
- **-change_host <OLD_HOST>**:- This is used when the files are moved to a new host or the name of the host is changed. All file paths lists that are from the old host will be updated with the name of the new host and the current path. If this option is omitted, the old location will be kept and a duplicate location will be created with the new host name.

6.1.1. Examine the necessary updates

It is highly recommended to check what the changes that going to take place after the update process. After the diff operation explained below a set of files will be created with the summary of details, full list of changes and the templates that will be used to make changes to the EGDM.

- **egenv -diff -configout <OUT_DIR>**
 - This perform a check for the files in the current directory against what is reported in the server. the -configout is a compulsory argument which points to a wright accessible folder to keep the reports and templates created. e.g. **egenv -diff -configout /home/myname/tmp/**
- **egenv -diff -recurse -configout <OUT_DIR>**
 - When the -recurse instruction is used not only the files in the current folder, but also all files in all subfolders will be checked.

6.1.2. Perform the updates

After checking the reports from the diff operations above, the updates could be commit in two ways.

- A. Use the templates created during the diff operation to perform updates one by one.
- B. Use the Auto mode described below to commit all updates are once.

- **egenv -update -current -recurse -configout <OUT_DIR>**

- When this command is used, user will not be warned of any change, instead all changes will be committed automatically.
- When the file checksum is locally changed (file modified after depositing), the update will change the checksum recorded for the file in the database .
- When a duplicate files is found (the path recorded in the EGDMS exists and in addition a new path is found). A new files2path relationship will be created
- When files are move (The location mentioned in the server is no longer available and instead a new location was detected). The old files2path relationship will be deleted and a new one would be created.

- **egenv -update --changehost <OLD_HOST> -current -recurse -configout <OUT_DIR>**

- When the **-changehost** instruction is used followed by the address/name of the old host, all locations which has an equivalent checksum in the current location will be updated.

6.2 Updating a specific table.

This example demonstrates how to change the description of a report named report_51. Note the *egenv* is called with the *-template* flag. If this flag is omitted a auto updated will be started instead (section 5.1).

- **egenv -update**

The above command will take you through a step-by-step process to construct a template that can used for the update.

```
egenv -prepare -update -template
```

```
Select the table to update.
```

```
0) address
1) donordetails
2) files
3) person
4) report
5) report_batch
6) report_batch
7) report_batch
8) report_batch
9) sampledetails
10) sampledetails
: 4
```

```
Select the column in the table report to search (to find the records to be updated)
```

```
0) description
1) entryDate
2) genomicReferenceReport_id
3) id
4) name
5) report_batch_id
6) report_type_id
7) reporter_id
: 4
```

```
Enter a search term for the column in name the table report find the record to be updated (type * to get all): report_51
```

```
File crated .egenvar_describe_report_name_report____.txt
```

```
Number of matches = 1. Written to file: .egenvar_describe_report_name_report____.txt
```

The new description should be added to the `.egenvar_describe_report_name_report____.txt` file. When editing make sure that tab is used only to separate columns and each line should end with a new line. Use a plain text editor and if you use word processors or spreadsheet editors make sure to save as plain text UTF-8. The issue the follwing command to update the database.

- **egenv -update <template_file>**
- For the above example this would be
 - **egenv -update .egenvar_describe_report_name_report____.txt**

7. Delete

Deleting records has two steps.

1. Prepare a template

- **egenv -prepare -delete**

OR

- **egenv -template -exact -search <TABLE>.<FIELD>=<search_value>** (please refer section 5 Search for search query construction details). **Warning do not use -expand** instruction here, if used there might be recursive deletes with unintended consequences.

2. Delete using the template

- **egenv -delete <TEMPLATE_FILE_NAME>**
- The template file name is the full name of the file created during the prepare step.
- Templates created for the *update* operation can also be used for the deletion step.

To delete the content of the current directory from the database, following command can be used.

- A. **egenv -delete -current**

- B. **egenv -delete <TEMPLATE_FILE_CREATED_IN_STEP_A>**

When the -delete instruction is used without the **-prepare** and with **-current** instruction, a template file is created by including the files in the current working directory (or when *in* argument is used, the user specified directory).

8. Advanced commands

8.2 Force the output directory*

The prepare files, templates and other files generated by the `egenv` is stored in the current and the parent directories. When there is no write permission for the dipositor on those directories the output could be forced to a different directory. To achieve this use the `-configout` instruction immediately followed by the output location. Please use the default directories whenever possible to maximize the automation process.

- `egenv -prepare -configout <out_put_directory>`
 - e.g. `egenv -prepare -configout /home/tmp/`

8.2 Force the input directory for config files*

When the output is forced to a different directory than the default, then this directory must be specified for `-add` and `-update` operations. This is done using the `-configin` flag.

- `egenv -add -configin <in_put_directory>`
 - e.g. `egenv -add configin /home/tmp/`

8.3 Force the input directory*

The default input directory or the directory to be processed is the current working directory. To override this use `in` instruction.

- `egenv -add -in /home/new/addfiles/`
 - The above example process the `/home/new/addfiles/` directory instead of the current working directory.

8.4. Override warnings

When file entries, the file path, server and read access will be checked. The `-override` instruction will prevent this check. This instruction could be used, when the files are not accessible during the depositing process.

8.5. Check changes

The -diff instruction could be used to check for changes in the files after the depositing process. This could also be used to list all content of a report when the report content is distributed across multiple locations.

8.6. Using egenv to check local file duplicates

The tool can be used to find out duplicates in a local drive without adding any information to the database.

- **egenv -prepare -clean -skipdb**
 - Create a report (.egenvar_describe_files_warnings.txt) with information on duplicates found using checksum and file name.
- **egenv -prepare -clean -skipdb -skipsize 200**
 - Check the local files as the above command, but ignore files larger than 200Mb

8.5. Change query timeout.

A query timeout value is set for webservices. This is to release frozen threads and to avoid long running process from occupying system resources. The default value for this is 300 (which will set a limit of roughly 6.5 minutes). If you set this to 1 then the timeout value will be between 1 to 2 minutes. If you want to force a query taking a longer time you could change this value.

The example below shows how to change the web services query timeout value using the template-update method as described under the Update section.

- **egenv.sh -add -split_limit 2000 -interval 60000 -template *txt> sampl_tagging.tag**
 - In this example all templates in the current directory and has the extension txt is loaded to the EGDMS. This scenario is expected to perform about a million changes and may take a long time.
 - The -split_limit 2000 sets the amount of records sent via web service at one instance. The default value for this 500, i.e. if you have more than 500 records in the template file, it will be split into chunks of 500 and sent to the EGDMS. in addition this value determines the UNDO resolution. i.e. when the value is set to 2000, after each update of 2000 an UNDO point is created. For the above example if we had one million changes to the database there will be 500 UNDO points.
 - the -interval 60000, sets the query timeout to about 10 minutes. thus even if the chunks of 2000 takes more than 10 minutes to finish, the operation will not be terminated. however this will bind resources for longer (others may face slower response)

#The commands and values the user needs to type are shown in **blue**

egenv -prepare -update

Select the table to update.

- 0) ADDRESS_TAGSOURCE
- 1) BIOLOGICAL_ONTOLOGIES_TAGSOURCE
- 2) CONFIG

- 3) DONORDETAILS
- 4) DONORDETAILS_HIERARCHY
- 5) ERRORMSGS

...

..

: **2**

Select the column in the table CONFIG to search (to find the records to be updated)

- 0) DESCRIPTION
- 1) DISPLAY
- 2) ID
- 3) NAME
- 4) PARAM_VALUE
- 5) URL=NA
- 7) Cancel

: **3**

Enter a search term for the column in NAME the table CONFIG find the record to be updated.

(type * to get all, close query within quotes (") to perform exact match):

WEBSERVICE_TIMEOUT

(2) Crating file .egenvar_describe_CONFIG_WEBSERVICE.txt @2014-02-18 14:49:01

Processing complete. Use "-update .egenvar_describe_CONFIG_WEBSERVICE.txt" to commit the chages to the database.

Or Use "-delete .egenvar_describe_CONFIG_WEBSERVICE.txt" to delete these from the database

#Then edit the above created template file .egenvar_describe_CONFIG_WEBSERVICE.txt to change the value of the CONFIG.PARAM_VALUE column. Then issue the update command as shown below to update the values in the database.

egenv -update .egenvar_describe_CONFIG_WEBSERVICE.txt

9. Error handling and debugging

9.1. Working with Write protected directories

The egenv tool requires a output directory to write the config files during some operations. When a write protected directory is found the user may be warned of this and a alternative location may be requested. However, during non-interactive batch mode, the tool may use the current directory, if that is writable, instead of waiting for the user. When this happens the following errors are be thrown

A. When trying to add files, after performing a prepare , where the parent directory is write protected (the default location of the report_batch config file).

Error 28c: Description files not found

"/home/sabryr/tmp/rand/.egenvar_describe_report_batch.txt". Please perform prepare before adding. (egenv --prepare)

if the config files are placed in a different location specify this with the -configin instruction

To get around this force the input location. using the following format

- **egenv -add -configin <full path for the directory where the config files are>**

9.2 Errors related to processing large amount of data or long running process.

By default the system is designed to terminate a process if the communication with the server failed to return a result within 250 seconds (about 4 minutes). This value is decided considering the maximum split load of 500. i.e. if there are more than 500 processing requests, it will be automatically split up. If you override this split value (e.g. **-split_limit 10000**) or if there is a process that you know would take a longer time than this then it is possible to override this timeout value. To increase the timeout use the following additional instruction with the command.

- **-interval <interval_in_seconds>**

e.g to increase this value when adding files

- **egenv.sh -split_limit 10000 -interval 15000 -add -configin file_det..txt**

If the timeout exceeded the processing will be terminated with the following error

- **Maximum waiting time exceeded**

Function	Command	description
Authenticate	egen -authenticate	Authentication is per location per user. If the software is installed on a common location, each user needs to be authenticated. The software cannot be used as the root user. A user authenticated already on a computer needs re-authorization if tried to access from a different computer
Prepare for adding	egen -prepare	This command will create three template files with descriptions of each file, the report and the batch. The command will prompt the user to accept a default value or modify the content on each step. By default, this will use the directory name as the report name and the parent directory name as the batch name. After creating the template, the property files could be edited using a plain text editor or using scripts to change the content. A log file named warnings will be created with the errors or problems occurred during the process. If used with the <i>-skipdb</i> option this could be used create a report about the duplicates after analysing all the subfolders from current location.
Add to the database	egen -add <files>	This will add details about a file or a set of files to the database. This command is dependent on the property files created by the -prepare option. If the report and the batch do not already exist they will be created. Before adding checksum values will be created for each files to be added and this and the canonical file path would be used to check for duplicates in the database.
Add relationships	egen -relatec <parent file> <child file/s>	This will add one or more files as child files for the parent files specified. When a file is used as input when creating another, the first one is the parent and the other is the child. Similarly six types of relationships are defined and described in supplementary

		material.
list	egen -l	List all available files.
Search	egen --search files.name= <file_name>	This will find the specific file specified. It is also possible to list files using wildcards and with more parameters providing fine tuning (supplementary material).
Debugging and repair	egen -diagnose	Check for errors in the installation and performs some maintenance.
Update software	egen -update	Update the software to the latest version
Help	egen -help	Display help information for available options.

10. Getting help from UNIX commands when automating

10.1. Transforming a text file to deed in as search parameter.

This example shows how could UNIX commands could be used to transform a file to the correct format to be used in a tag based search.

ProjectID	PID@104555	SentrixBarcode	SentrixPosition
000388	1045550292122	4515597045	R01C01
000388	1045550667913	4515597045	R02C01
000388	1045550442282	4535791163	R01C01
000388	1045550660020	4535791163	R01C02
000388	1045550349284	4535791163	R02C01
000388	1045550321945	4535791163	R02C02
000388	1045550535441	4535803076	R01C01
000388	1045550058117	4535803076	R01C02

Box 10.1.1 (original.txt)

In the extract 10.1.1, a section of the original files is shown. This has four columns and only the last two will be used in this particular instance. Moreover, the SentrixBarcode is parent term and SentrixPosition is the child term (see tag based search for more details). When the command below is executed the file shown in 10.1.2 is created.

```
cat original.txt | awk -F'\t' '{print $3,".", $4}' | sed -e 's/ //g' > transformed.txt
```

*Split each row using the tab character (-F\t) then take the columns 3 and 4 and combine it with a "." ('{print \$3,".", \$4}'). Remove all the blank spaces (sed -e 's/ //g'). Create a new file with this extracted information (> transformed.tx)

SentrixBarcode.SentrixPosition
4515597045.R01C01
4515597045.R02C01
4535791163.R01C01
4535791163.R01C02
4535791163.R02C01
4535791163.R02C02

Box 10.1.2 (transformed.txt)

The transformed file can be used in the search as follows,

- **egenv -search -exact -file TAG=transformed.txt sampledetails -out samples.txt**

*Use the content to search through files and find the corresponding sampledetails. Save this information in the files named samples.txt

10.2. Other helpful commands

- `awk '{ print $5 }' egenvar_files.txt`
 - Print the fifth column of the file `egenvar_files.txt`
- `egenv.sh -dontformat -exact -search tag=4535811005.R02C02 "sampledetails|files2path.filepath" | grep idat | tar cvzf zipped.tar.gz -T -`
 - Find all the files which are attached to sample details tagged with "4535811005.R02C02" and then compress those to the file "zipped.tar.gz"