Tutorial on X-DB System (Version 0.3.1)

Table of Contents

Introduction.

This is the third release of the X-DB System tutorial. In this tutorial I'll explain the basic concepts of the system, the commands and the web access to it.

General concepts

The system is installed in a server called x-db.iciq.es (10.3.1.210). To access the system using the shell you have to install in your Desktop computer some software and setup some environment variables.

For the moment there is only one role in the System: “users”, this role give access to all the operations you can do in the repository. In the future, there will be more roles and according to the role the user will be able to do something or not to do it.

The x-db system is accessed through bash command lines and you can access at the same time to your local hard disk and also to the remote system.

In your local hard disk, you can navigate using cd, ls, and so on. You have a “home”, and so on, and so on. In the repository you also have a “home”. In that case the “home” is /db/username, for example: /db/jiglesias. Into this structure, you can create and manage “projects” and “calculations”. Projects are the units that let you to group other projects and calculations. The control access to your data in the database is managed through the project concept. Calculation concept inherit the private policy of the parent project, but each project can have their own privacy policy. Later I'll show you some samples.

To load and manage files located in other machines others than your local machine, you will can create an special directory with all the remote file systems mounted there. For example, you will have a structure like this:

/home/mbesora/remote-machines/kimik
/home/mbesora/remote-machines/kimik2
/home/mbesora/remote-machines/marenostrum
/home/mbesora/remote-machines/cesca
...

Using the bash shell commands you can access those remote systems. How to mount those remote systems will be explained in another tutorial (the fuse one).

Shell Installation.

The first step is to install the repository client in your desktop computer. The shell works fine in Linux and Mac OS, but not in Windows.

Step by Step installation HOWTO for Linux (for MacOS you can go to step 3):

1. execute the command uname -a. If it's printed x86_64 you have a 64 bits architecture. In other case you have a 32 bits architecture.
   1. If 32 bits architecture:
      1. Download the Java Runtime Environment version 6 from the following link: http://www.oracle.com/technetwork/java/javase/downloads/index.html
         Choose the JRE column and download Linux x86 - Self Extracting Installer
   2. In the other case:
      1. Download the Java Runtime Environment version 6 from the following link: http://www.oracle.com/technetwork/java/javase/downloads/index.html
         Choose the JRE column and download Linux x64 - Self Extracting Installer
2. After downloaded in a temporal directory execute:
   1. chmod +x on the file
   2. execute the file. A directory structure will be uncompressed.
   3. Copy the uncompressed directory structure into a location of your home. For exemple at /home/user/bin.
   4. Edit the .bashrc file. And add:
      1. export PATH=path_to_the_bin_sub_directory_of_the_uncompressed_directory:$PATH
   5. After that step you have successfully installed the JRE environment necessary to execute Java programs.
3. Now is time to install the shell. The shell is composed of several files:
   1.
   2. Copy the following files into the /home/user/bin directory.
      1. exe-rep-command
      2. exit-rep
      3. start-rep-shell
      4. shell.jar
   3. cd into the bin directory and execute chmod +x over each file.
   4. Copy the cert.bin file into this directory (the bin one).
   5. Execute chmod 600 cert.bin
   6. Edit the .bashrc file and add the following variables:
      1. export USERDB="your_user_name"
      2. export REPPASS="your password"
      3. export CERT_FILE="/home/username/bin/cert.bin"
      4. export REP_SCRIPTS="/home/username/bin"
      5. export PATH=$REP_SCRIPTS:$PATH
      6. export REPOSITORY_IP="10.3.1.210"
4. The setup of the shell is done. For mounting remote homes, check the fuse tutorial.
5. To properly view the molecules in the web interface you have to make a soft link in the plugins directory of the firefox. The place of this directory depends on each distribution, but it uses to

be at /usr/lib. Once located the directory execute

```
 ln -s /path/to/java/run/time/environment/lib/processor_architecture/libnpjp2.so libnpjp2.so
```

*processor_architecture have to be i386 or amd64 or another similar name.

You need root permissions to do this. After this operation is done, you will be able to visualize molecules in your browser using the repository web interface

General shell commands overview.

The best way to learn how to interact with the system is through examples:

Command start-rep-shell
After the installation is done you have to start a session in the remote x-db system executing:

```
jiglesias@p008:~$ . start-rep-shell
Starting Repository Shell
[1]+  Done                    echo 'Starting Repository Shell'
```

Pay spacial attention to the "." plus an "space" before the start-rep-shell command.

After that a remote session is started.

Command pwdpro

Now for example you can pwd the remote system executing:

```
jiglesias@p008:~$ pwdpro
/db/jiglesias
jiglesias@p008:~$
```

Command lspro

You can browse the repository using lspro:

```
jiglesias@p008:~$ lspro
{proj001}   {proj002}   {proj003}   {proj004}
jiglesias@p008:~$
```

The symbol "{" and "}" means that concept is a "project", without "{" "}" means that it's a calculation.

If you want a more detailed list of the projects of a path:

```
jiglesias@p008:~$ lspro -f
Type  Name       Description        Permission  Owner      Group      Creation time       Concept Group       State
-------------------------------------------------------------------------------------------------------------------------
PRO   proj001    zsdf               111000      jiglesias   cbo       2010-09-27 20:37:40   general     created
PRO   proj002    asddddd            111000      jiglesias   cbo       2010-09-27 20:37:40   free        created
PRO   proj003    bgggfgf            111000      jiglesias   cbo       2010-09-27 20:37:40   potential      created
PRO   proj004    badfdfd dfd        111000      jiglesias   cbo       2010-09-27 20:37:40   potential+free    created
jiglesias@p008:~$
```

Here you have 4 projects, with name description, permissions, and so on. Concept Group can have one of the following values:

1. general: no concrete meaning aggregation of projects or calculations.
2. Potential: for potential energy surfaces
3. free: for free energy surfaces

4. Fermi_level
5. imaginary_frequency
6. binding_energy
7. of a “+” separated values like : potential+free

You can modify the order in which the projects are listed using the flag “order by” -o

For example:

```
jiglesias@p008:~$ lspro -o o
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group        State
------------------------------------------------------------------------------------------------------------------------
PRO   proj001    zsdf              111000     jiglesias   cbo       2010-09-27 20:37:40   general    created
PRO   proj002    asddddd             111000     jiglesias   cbo       2010-09-27 20:37:40   potential    created
PRO   proj003    bgggfgf            111000     jiglesias   cbo       2010-09-27 20:37:40   free      created
PRO   proj004    badfdfd dfd         111000     jiglesias   cbo       2010-09-27 20:37:40   potential+free    created
jiglesias@p008:~$
```

This command will list the contents of the project ordered by “owner”. This is done using -o o.
You have:

-o o : owner
-o n : name
-o g: group
-o t: creation time
-o c: concept group
-o s: state.

The permissions of a project are managed in a linux way. The first two bits means read and write of the owner. The third and forth bits means read and write for the group, the the last two means read and write for the other users.

For example lets to modify the permissions of the proj002:

```
jiglesias@p008:~$ mpro proj002 -p 110000
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group        State
------------------------------------------------------------------------------------------------------------------------
PRO   proj001    zsdf              111000     jiglesias   cbo       2010-09-27 20:37:40   general    created
PRO   proj002    asddddd             110000     jiglesias   cbo       2010-09-27 20:37:40   potential    modified
PRO   proj003    bgggfgf            111000     jiglesias   cbo       2010-09-27 20:37:40  free      created
PRO   proj004    badfdfd dfd         111000     jiglesias   cbo       2010-09-27 20:37:40   potential+free     created
jiglesias@p008:~$
```

As you can appreciate the permissions on proj002 have changed and also the state has changed from created to modified.

Command catpro

You can also cat a project:

```
jiglesias@p008:~$ catpro proj001
Printing the contents of proj001
--------------------------------
Name:proj001
Description:zsdf
Permissions:111000
Owner:jiglesias
Group:cbo
Path:/db/jiglesias
Concept Group:general
Creation Time:2010-09-27 20:37:40
Modification Time:null
Certification Time:null
State:created
jiglesias@p008:~$
```

Command cdpro

Lets to navigate into a project:

```
jiglesias@p008:~$ cdpro proj001
jiglesias@p008:~$ lspro -f
Type  Name      Description       Permission  Owner     Group     Creation time       Concept Group     State
--------------------------------------------------------------------------------------------------------------------
CAL   cal001    A description     111000      jiglesias  cbo      2010-09-27 20:37:40  general           created
CAL   cal002    A description     111000      jiglesias  cbo      2010-09-27 20:37:40  general           created
CAL   cal003    A description     111000      jiglesias  cbo      2010-09-27 20:37:40  general           created
CAL   cal004    A description     111000      jiglesias  cbo      2010-09-27 20:37:40  general           created
CAL   cal005    A description     111000      jiglesias  cbo      2010-09-27 20:37:40  general           created
CAL   cal006    A description     111000      jiglesias  cbo      2010-09-27 20:37:40  general           created
jiglesias@p008:~$
```

In this sample we have some calculations. And if you type:

```
jiglesias@p008:~$ lspro
cal001   cal002   cal003   cal004   cal005   cal006
jiglesias@p008:~$
```

You can see the calculations without {}.

To move one project down:

```
jiglesias@p008:~$ cdpro ..
jiglesias@p008:~$ lspro -f
Type  Name      Description       Permission  Owner     Group     Creation time       Concept Group     State
--------------------------------------------------------------------------------------------------------------------
PRO   proj001    zsdf            111000     jiglesias   cbo      2010-09-27 20:37:40   general     created
PRO   proj002    asddddd           110000     jiglesias   cbo      2010-09-27 20:37:40   potential     modified
PRO   proj003    bgggfgf           111000     jiglesias   cbo      2010-09-27 20:37:40   free      created
PRO   proj004    badfdfd dfd          111000      jiglesias   cbo        2010-09-27 20:37:40   potential+free     created
jiglesias@p008:~$
```

Don't use expressions like cdpro ../../something/something-else. That is not allowed. For the moment the command just use simple paths like .. , a name and also a long path /path/to/somewhere

Command cpro

Lets go to create a project:

```
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group        State
-----------------------------------------------------------------------------------------------------------------------------
PRO   proj001    zsdf               111000      jiglesias   cbo        2010-09-27 20:37:40   general          created
PRO   proj002    asddddd            110000      jiglesias   cbo        2010-09-27 20:37:40  potential     modified
PRO   proj003    bgggfgf            111000      jiglesias   cbo        2010-09-27 20:37:40   free      created
PRO   proj004    badfdfd dfd         111000      jiglesias   cbo        2010-09-27 20:37:40   potential+free    created
jiglesias@p008:~$ cpro -n myfirstpro -d Some destription -cg potential
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group        State
-----------------------------------------------------------------------------------------------------------------------------
PRO   myfirstpro  Some destription      111000      jiglesias   cbo        2010-09-28 15:33:24   potential  created
PRO   proj001    zsdf               111000      jiglesias   cbo        2010-09-27 20:37:40   general     created
PRO   proj002    asddddd            110000      jiglesias   cbo        2010-09-27 20:37:40   potential     modified
PRO   proj003    bgggfgf            111000      jiglesias   cbo        2010-09-27 20:37:40   free      created
PRO   proj004    badfdfd dfd         111000      jiglesias   cbo        2010-09-27 20:37:40   potential+free     created
jiglesias@p008:~$
```

Don't use very long name for the projects.

Once you set a name very long for a project, maybe using lspro -f you will not see the entire name, in that case execute lspro without arguments, and you will see the name as long as it is. But try to don't use very long names. The usual length should be 10 characters.

Command mpro

If you want to modify a project use the command: mpro.

```
jiglesias@p008:~$ lspro -f
Type  Name       Description          Permission  Owner      Group      Creation time        Concept Group        State
-----------------------------------------------------------------------------------------------------------------------------
PRO   myfirstpro  Some destription      111000      jiglesias   cbo        2010-09-28 15:33:24  potential  created
PRO   proj001     zsdf                 111000      jiglesias   cbo        2010-09-27 20:37:40   general     created
PRO   proj002     asddddd              110000      jiglesias   cbo        2010-09-27 20:37:40   potential     modified
PRO   proj003     bgggfgf              111000      jiglesias   cbo        2010-09-27 20:37:40   free      created
PRO   proj004     badfdfd dfd          111000      jiglesias   cbo        2010-09-27 20:37:40   potential+free     created
jiglesias@p008:~$ mpro myfirstpro -d Another description
jiglesias@p008:~$ catpro myfirstpro
Printing the contents of myfirstpro
-----------------------------------
Name:myfirstpro
Description:Another description
Permissions:111000
Owner:jiglesias
Group:cbo
Path:/db/jiglesias
Concept Group:potential
Creation Time:2010-09-28 15:33:24
Modification Time:2010-09-28 15:39:00
Certification Time:null
State:modified
jiglesias@p008:~$
```

You can always ask for help in a command typing: command -h. For example:

```
jiglesias@p008:~$ mpro -h
mpro
mpro modification of the project
Options:
        Option -d:  Description of the project                (optional)
        Option -nn:  New Name of the project                  (optional)
        Option -np:  New Parent project (absolute path)        (optional)
        Option -n:  Relative or absolute project path          (mandatory)
        Option -cg:  Concept Group of the project               (optional)
        Option -p:  Permissions of the project. Ex: '110100'    (optional)
        Option -o:  Owner of the project                      (optional)
        Option -g:  Group owner of the project                (optional)
jiglesias@p008:~$
```

The -n flag can be usually omitted and usually taken from the first parameter. Example:

```
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group       State
---------------------------------------------------------------------------------------------------------------------------
PRO   myfirstpro  Another description   111000     jiglesias   cbo        2010-09-28 15:33:24   potential  modified
PRO   proj001     zsdf               111000     jiglesias   cbo        2010-09-27 20:37:40   general     created
PRO   proj002     asddddd            110000     jiglesias   cbo        2010-09-27 20:37:40   potential     modified
PRO   proj003     bgggfgf            111000     jiglesias   cbo        2010-09-27 20:37:40   free      created
PRO   proj004     badfdfd dfd        111000     jiglesias   cbo        2010-09-27 20:37:40   potential+free     created
jiglesias@p008:~$ cdpro proj001
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group       State
---------------------------------------------------------------------------------------------------------------------------
CAL   cal001     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal002     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal003     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal004     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal005     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal006     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
jiglesias@p008:~$
```

Is the same as:

```
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group       State
---------------------------------------------------------------------------------------------------------------------------
PRO   myfirstpro  Another description   111000     jiglesias   cbo        2010-09-28 15:33:24   potential  modified
PRO   proj001     zsdf               111000     jiglesias   cbo        2010-09-27 20:37:40   general     created
PRO   proj002     asddddd            110000     jiglesias   cbo        2010-09-27 20:37:40   potential     modified
PRO   proj003     bgggfgf            111000     jiglesias   cbo        2010-09-27 20:37:40   free      created
PRO   proj004     badfdfd dfd        111000     jiglesias   cbo        2010-09-27 20:37:40   potential+free     created
jiglesias@p008:~$ cdpro -n proj001
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group       State
---------------------------------------------------------------------------------------------------------------------------
CAL   cal001     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal002     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal003     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal004     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal005     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
CAL   cal006     A description       111000     jiglesias   cbo        2010-09-27 20:37:40   general               created
jiglesias@p008:~$
```

Command dpro:

Lets go to delete a project:

```
jiglesias@p008:~$ pwdpro
/db/jiglesias
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group        State
------------------------------------------------------------------------------------------------------------------------------
PRO   myfirstpro  Another description   111000     jiglesias   cbo        2010-09-28 15:33:24   potential  modified
PRO   proj001    zsdf              111000     jiglesias   cbo        2010-09-27 20:37:40   general    created
PRO   proj002    asddddd            110000     jiglesias   cbo        2010-09-27 20:37:40   potential    modified
PRO   proj003    bgggfgf            111000     jiglesias   cbo        2010-09-27 20:37:40   free     created
PRO   proj004    badfdfd dfd         111000     jiglesias   cbo        2010-09-27 20:37:40   potential+free    created
jiglesias@p008:~$ dpro myfirstpro
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group        State
------------------------------------------------------------------------------------------------------------------------------
PRO   proj001    zsdf              111000     jiglesias   cbo        2010-09-27 20:37:40   general    created
PRO   proj002    asddddd            110000     jiglesias   cbo        2010-09-27 20:37:40   potential    modified
PRO   proj003    bgggfgf            111000     jiglesias   cbo        2010-09-27 20:37:40   free     created
PRO   proj004    badfdfd dfd         111000     jiglesias   cbo        2010-09-27 20:37:40   potential+free    created
jiglesias@p008:~$
```

Be aware!!! Only it's checked the project permissions you want to delete and are applied recursively to all subproject at delete time. A deletion is **always** recursive.

Command findpro

Maybe you would like to find some string or regular expression on some fields of the projects. For this purpose you will have findpro command. You can find which project contains certain text value in their fields.

For example:

```
jiglesias@p008:~$ lspro -f
Type  Name       Description         Permission  Owner      Group      Creation time        Concept Group        State
------------------------------------------------------------------------------------------------------------------------------
PRO   proj001    zsdf              111000     jiglesias   cbo        2010-09-27 20:37:40   general    created
PRO   proj002    asddddd            110000     jiglesias   cbo        2010-09-27 20:37:40   potential    modified
PRO   proj003    bgggfgf            111000     jiglesias   cbo        2010-09-27 20:37:40   free     created
PRO   proj004    badfdfd dfd         111000     jiglesias   cbo        2010-09-27 20:37:40   potential+free    created
jiglesias@p008:~$ findpro -h
findpro
findpro description. Always searches in a recursive way.
Options:
          Option -d:  Regular expression to find in the description field of project   (optional)
          Option -n:  Regular expression to find in the name field of project         (optional)
          Option -p:  Regular expression to find in the path field of project         (optional)
```

```
jiglesias@p008:~$ findpro -d zsd
Name            Path
--------------------------------------------------------------------------------
proj001         /db/jiglesias
jiglesias@p008:~$ exit-rep
jiglesias@p008:~$
```

The command just searches in the current path a description containing zsdf

In the next example I create a project into proj002.

Then I search specifying the absolute path where I want to search. Lets see:

```
jiglesias@p008:~$ cdpro proj002
jiglesias@p008:~$ lspro

jiglesias@p008:~$ cpro test -d desci -cg gggg
jiglesias@p008:~$ lspro
{test}
jiglesias@p008:~$ lspro -f
Type  Name      Description       Permission  Owner      Group      Creation time       Concept Group      State
-------------------------------------------------------------------------------------------------------------------------
PRO   test     desci           111000     jiglesias   cbo      2010-09-28 20:58:56   gggg           created
jiglesias@p008:~$ cdpro ..
jiglesias@p008:~$ lspro -f
Type  Name      Description       Permission  Owner      Group      Creation time       Concept Group      State
-------------------------------------------------------------------------------------------------------------------------
PRO   proj001    zsdf            111000     jiglesias   cbo      2010-09-27 20:37:40   general    created
PRO   proj002    asddddd          110000     jiglesias   cbo      2010-09-27 20:37:40   potential    modified
PRO   proj003    bgggfgf          111000     jiglesias   cbo      2010-09-27 20:37:40   free     created
PRO   proj004    badfdfd dfd       111000     jiglesias   cbo      2010-09-27 20:37:40   potential+free    created
jiglesias@p008:~$ findpro /db/jiglesias/proj002 -d desc
Name            Path
--------------------------------------------------------------------------------
test            /db/jiglesias/proj002
jiglesias@p008:~$ findpro /db/jiglesias -d desc
Name            Path
--------------------------------------------------------------------------------
test            /db/jiglesias/proj002
jiglesias@p008:~$
```

Command loadcalc

And finally we arrive at the loadcalc command. This command load the contents of a calculation file or files into the repository. For this purpose uses a xml definitions placed at: http://datachem.iciq.es/xml/extraction/cutting-areas/

With ADF, Gaussian and VASP definitions. Those definitions were created by special users of this room called data-architects. Those files define using regular expression and xml tags what information to cut in your calculations, and that cut text is associated with a meaning name. A meaning for the moment is also called Area (A peace of text). But in the future will be a very concrete data (for example potential-energy and the value of it).

Lets to see a sample:

Into the same directory we have two gaussian outputs:

```
jiglesias@p008:~/test/extraction-tests/test002$ ls -l
total 480
-rw-r----- 1 jiglesias jiglesias 244651 2010-09-16 15:24 gaussian2.out
-rw-r----- 1 jiglesias jiglesias 244650 2010-09-14 17:31 gaussian.out
-rw-r----- 1 jiglesias jiglesias 244650 2010-09-14 17:31 gausinput.in
jiglesias@p008:~/test/extraction-tests/test002$
```

If we just execute:

```
jiglesias@p008:~/test/extraction-tests/test002$ loadcalc test -d some description
Type is : Gaussian
jiglesias@p008:~/test/extraction-tests/test002$
```

In that case the first .out file and the first .in file, or the VASP special ones are searched and loaded automatically. The command prints the calculation type detected.

Imagine you want to specify a concrete output and input file to load, then you can execute:

```
jiglesias@p008:~/test/extraction-tests/test002$ loadcalc -o gaussian.out -i input.in
Type is : Gaussian
jiglesias@p008:~/test/extraction-tests/test002$
```

You can also specify the calculation type using the flags:

```
jiglesias@p008:~/test/extraction-tests/test002$ loadcalc -og gaussian2.log  -ig input.in
Type is : Gaussian
```

Type loadcalc -h for the complete flag options, and other type calculation flags.

Command loadvasp

This command is an improvement of loadcalc for vasp calculations. It loads the vasp calculation faster if it's executed into a directori with the default vasp file names.

Ex: loadvasp a_name -d a description

If the contents of the directory is the usual one in vasp the command automatically will load the vasp calculations without any other parameter.

Command loadneb

This command is another special command for vasp. It loads a NEB. Type loadneb -h for more help, a complete help will be printed. The command automatically creates a project to group the initial state calculation, the transition state calculations, and the final state calculation.

Command loadidm

This is the last vasp special command. It loads IDM calculations in a similar way of loadneb, because it creates a project containing the initial state calculation, transition state calculation and the final state calculation. Type loadidm for more help.

Command viewcalc

To view the calculation loaded into the repository execute this command.

Example:

```
jiglesias@p008:~/test/extraction-tests/test002$ viewcalc test
Printing the contents of test
-----------------------------
Name:test
Description:dddddd
Permissions:111000
Owner:jiglesias
Group:cbo
Type:VSP
Path:/db/jiglesias/proj002
Concept Group: potential
Creation Time:2010-10-12 21:32:47.473975
Certification Time:null
jiglesias@p008:~/test/extraction-tests/test002$
```

Here we are checking the basic contents of the calculation stored in the database.

To check the whole data stored in the database execute:

```
 viewcalc test -f
```

This command will print the contents of the basic data, plus the contents of the Areas. Here goes a sample of the printing of a some Areas:

```
<<<k-points used in calculation and their position.>>>
[[[Monkhorst Pack
0
Gamma
 1  1  1
 0  0  0

]]]
<<<Compute Engine-Extraction.>>>
[[[ vasp.4.6.21  23Feb03 complex
]]]
```

The format of the contents areas is:

<<< name of the area >>>
[[[the contents of the area]]]

If the calculation has Areas in the form of files stored in the database, those files will be download in the current path. The loadcalc command stores areas in the form of file if the area is bigger than 3Kb or information of the final geometry.

Command exit-rep

Don't forget to execute:

exit-rep to exit the remote x-db system.

We interface .

There is also a web interface to make most of the operations. You have a tutorial in the wiki called x-db-web.pdf