Grid Commands Usage with examples

Author: Vanamala Venkataswamy
Date: 01/17/2012

Contents

1. Introduction

Genesis II is a grid software system designed to provide both compute and data grid functionality to end- users. Using GenesisII, a user can just as easily run one job as he or she can run a hundred, or a thousand. A user can copy data into the grid or out of the grid and can mirror existing data and data directories or folders into the grid for ease of access later or for use as a collaboration mechanism with other researchers across the globe. In essence, the grid allows users to use and share network connected resources just as easily as if they were all contained in a single, extremely powerful computer sitting on the user's desktop.

To be able to use this guide, user should have already downloaded and installed GenesisII client software. Once installed, user can invoke grid shell either from command line directly or from client-ui.

Eg: Command line

```
cd $GENII_INSTALL_DIR/grid shell
```

or

client-ui → Tools → Launch grid shell

2. List of Commands

Once you are in grid shell, type 'help' to get a list of all the commands/tools that you can use.

```
grid help
```

Each command also has man page and help page. To see man page/help page of command, type this in grid shell

```
grid man command
```

```
grid help command
```

Following is the complete list of commands available. Some of the commands need administrative privileges to execute, these commands are useful only for Grid administrators.

```
grid help
```

Administration
- attach-host
- bes-manager
- bes-policy
- get-attributes
- get-bes-attributes
- matching-params
- mint-epr
- set-container-service-properties

Data
- cat
- cd
- cp
- edit
- export
- fuse
- ln
- ls
- mkdir
- mv
- rm
- unlink

General
- client-ui
- history
- ping
- pwd
- schedule-termination
- script
- set
- set-resource-properties
- set-user-config
- update
- user-preferences

Internal Use
- GetUserDir
- cloudTool
- container-stats
- idp

Job/Execution
- job-tool

parseJSDL
qcomplete
qconfigure
qkill
qlist
qquery
qreschedule
qstat
qsub
resource-history
run
runJSDL

Misc

connect
create-resource
echo
tty

Security

IDPLogin
authz
cert-generator
chmod
create-user
create-user-delegate
keystoreLogin
login
logout
passwordLogin
shell-login
whoami

3. Administration Tools

As the name specifies, these are mostly used by Grid administrators to add new resources to grid, to set/change properties of existing resources and to manage the resources in grid.

attach-host

The attach-host tool allows a user to attach a container into a different hosting environment. The container is specified by the container-URL The name of the container in the new hosting environment is specified by hosting-environment-rns-path.

To Attach new container to your RNS path

```
grid attach-host https://hostname:18080/axis/services/VCGRContainerPortType /home/vana/newContainer
```

18080 is the default port number, this can be configured to any port number while installing the container.

Bes-manager

The bes-manager tool provides a convenient (GUI) way for creating BES' on the container deployed on the same machine as the client. The BES can be linked into the grid name-space. This permits CPU cycle-sharing. The BES-Manager tool provides options for choosing whether to continue (or suspend or kill) running jobs on a particular BES (created using the BES-Manager tool), when the screen-saver is inactive or when a user is logged-in. The screen-saver inactive action and user logged in action for a BES, can also be set and queried using the bes-policy command.

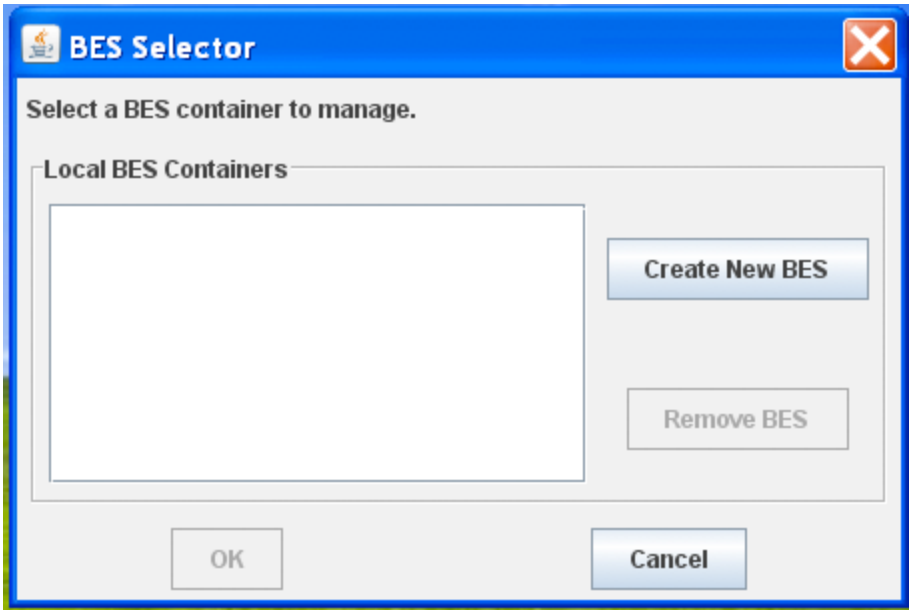

Figure.1 Initial BES Manager GUI

To add a new BES container to the XCG (i.e., to start supplying compute resource for jobs), click on the "Create New BES" button. This will bring up the dialog box shown in Figure 3 that asks for a grid RNS path (not a local directory path) where the BES container should reside. This path is necessary as it provides a mechanism for referring to the new BES container inside of the grid. The information provided here will need to be given to an administrator later.

Note that in order for this step to succeed, the XCG grid must be up and running and you must be able to connect to it from your computer (i.e., you must have network support). Further, you must have grid permissions to write into the directory you specify. Generally speaking, the best place to link the BES container into the grid is inside of a grid directory located somewhere at or below your home directory in the grid.

Figure.2 RNS path for new BES

Once you have the grid directory named, clicking on OK will submit the request to the XCG and will perform the necessary operations to make the new BES available through the grid name-space Figure 3 shows this final screen.

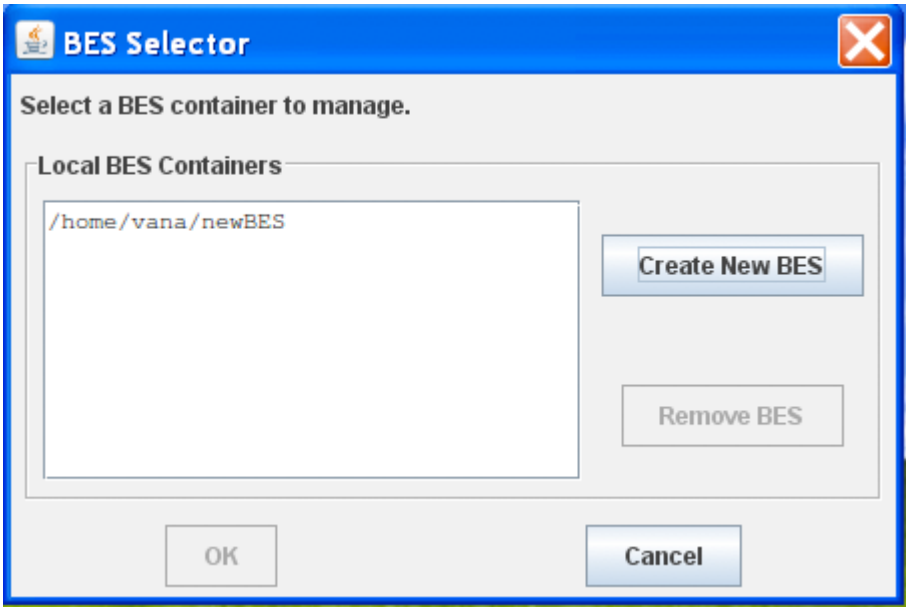

Figure. 3 Final screen BES creation

At this point, the BES container has been created and linked into the path that you specified, but the container is not yet generally available for jobs to run on. That final step requires the intervention of an XCG administrator who will perform the necessary steps to put it into the grid queue and give the container the necessary access control lists. Once you have created the container, please email the XCG group at xcg-help@virginia.edu and request that your BES container be added into the grid. At a minimum, please supply the following information:

- Your name and XCG grid account path
- Where the machine is located (e.g., “my desk in Olsson Hall in the Computer Science Department at the University of Virginia”)
- The path that you linked the BES container into
- How many slots (jobs) you would like to make available to the grid at any given time
- Whether or not you require that jobs be killed on the machine when you log in to your machine

bes-policy

BES Policies are rules or conditions that, when met, trigger some action to be taken with activities currently managed by a given BES container. Usually the rules have to do with an idle resource no longer being idle (i.e., now in use by a person). Actions range from doing nothing, to suspending, or even killing the activities on that container.

To Query the properties of a BES

```
grid bes-policy --target-bes /home/vana/myBES --query
```

To set properties of the BES

```
grid bes-policy --no-gui --target-bes /home/vana/myBES --set-user-logged-in=NOACTION -set-
screensaver-inactive=NOACTION
```

```
grid bes-policy --no-gui --target-bes /home/vana/myBES  --set-user-logged-in=SUSPEND --set-screensaver-inactive=SUSPEND
```

get-attributes
The get-attributes command, retrieves and prints the attribute (resource-property) document for a target.

To get attributes of a file

```
grid get-attributes /home/vana/testFile.txt
```

To get attributes of a directory

```
grid get-attributes /home/vana/testDir
```

get-bes-attributes
The get-attributes command, retrieves and prints the attribute (resource-property) document for a target.

To get BES attributes

```
grid get-attributes /home/vana/myBES
```

matching-params
This command is used to add/remove matching-parameters to a resource. This command is particularly useful when dealing with BES resources. If the "requires" property-identifier is used, then only jobs whose requirements match with those (required) properties of the BES, are scheduled onto that BES. In other words, each of the (required)BES property must be a requirement for the job too. If the "supports" or no requirement is used, then those BES properties need not be a requirement for the job. The matching-parameters for a BES resource can be retrieved using the get-bes-attributes command.

To add support for MPI on a BES, if user submits MPI job it will be matched to this BES

```
grid matching-params BES 'add(support-mpi,true)' and 'add(requires-mpi,true)
```

To remove support for MPI on a BES, if user submits MPI job it will NOT be matched to this BES

```
grid matching-params BES 'remove(support-mpi,true)' and 'remove(requires-mpi,true)
```

mint-epr
This command is used to generate epr for any resource.

To mint an epr for a resource and store the epr file in eprFile

```
grid mint-epr grid:/home/vana/newResource --output=/home/vana/eprFile
```

Link the BES into the Genesis II name-space

```
grid mint-epr --link=<bes-gffs-path> --certificate-chain=local:<path-to-container-cert> <bes-URL>
```

where the bes-URL has the following form https://hostname:port/deployment-name/services/GeniiBESPortType

set-container-service-properties
This command is used to set container services properties like download-manager temporary directory and the scratch space directory paths.

To set container properties Download directory. The path to which the download-manager-temporary-directory-path needs to be changed to. It is in the local file-system name-space The default location is the download-tmp directory in the GenesisII state directory.

```
grid set-container-service-properties --download-mgr-tmpdir=/tmp/Download
```

The path to which the scratch-space-directory-path needs to be changed to. It is in the local file-system name-space The default location is the scratch-space directory in the GenesisII state directory.

```
grid set-container-service-properties --cratch-space-dir=/tmp/Scratch
```

4. Data Tools

These tool are used to perform file system like operations on resources on grid and local name-space To be able to successfully use the commands, user should have appropriate permissions on that resource.

cat
The cat command, similar to cat in UNIX, reads the contents of specified ByteIO service resources and local files and displays their contents to the standard output stream.

Suppose you have 2 files, testFile1.txt and testFile2.txt

To cat a file in grid name-space

`grid cat /home/vana/gridFile1.txt` OR `gridcat grid:/home/vana/griFile1.txt`

To cat multiple files in grid name-space

```
grid cat /home/vana/gridFile1.txt /home/vana/gridFile2.txt
```

To cat a file on your local disk

```
grid cat local:/localdir/localFile3.txt
```

Note: you have to specify local: to be able to access local file system files

cd
Genesis II grid clients maintain a notion of current working RNS. This is the path from a root with EPRS of a name-space implemented with RNS that functions similarly to a file system. The cd tool allows a user to explicitly change his or her location inside that name-space

To cd to newDir

```
grid cd /home/vana/newDir
```

cd using .. like in *nix systems

```
grid cd ../
```

cp
Similar to cp in the UNIX operating system, the grid cp command copies the contents of one source file or ByteIO into another target file or ByteIO. If the target path specified does not exist, a new file or ByteIO resource is created. If a new ByteIO resource needs to be created in the grid environment, then the default ByteIO service on the default container is used to create it. Currently, there is no way to override this behavior without using the create-resource tool.

To copy one file to another

```
grid cd /home/vana/gridFile1.txt /home/vana/gridFile2.txt
```

To Copy grid file to local file

```
grid cd /home/vana/gridFile1.txt local:/localdir/localFile1.txt
```
OR
```
gridcd grid:/home/vana/gridFile1.txt local:/localdir/localFile1.txt
```

To copy local file to grid file

```
grid cd local:/localdir/localFile1.txt grid:/home/vana/gridFile3.txt
```

edit
Used to edit a file. If the file does not already exist, it will be created. In the grid-namespace this file will be created on the same container as its parent directory. The created file is a RandomByteIO resource.

```
grid edit /home/vana/ex.txt
```

export
The export tool gives users the ability to create and terminate grid exports of existing data (please reference exporting for more information on exports in the grid). If no command line values are given, then export launches a graphical dialog to help maintain local exports. Note that the dialog only allows users to manipulate exports created through the dialog (in other words, one cannot create an export on the command line, and then terminate it through the dialog) and only works in environments where a container has been started (i.e., you cannot use the export dialog to create exports on a remote machine).

To export a local directory to grid name-space

```
grid export --create ${containerPath}/Services/LightWeightExportPortType local:/local/exportDir
$/home/vana/exportDir
```

To quit exported directory, the exported directory will no longer be available in grid name-space

```
grid export --quit $/home/vana/exportDir
```

fuse

The fuse tool is used to mount the GenesisII grid name-space directly into the local UNIX environment. The mounted file system works just like any other mount on the UNIX system. In addition to normal file and directory operations, the users can submit jobs into the grid by copying JSDL documents into BES containers or GenesisII Queue resources.

To fuse mount the grid and run it in background (if --daemon is not specified the shell hangs until fuse mount quits)

```
grid fuse --mount local:/localdir/mountDir --daemon
```

To fuse mount the grid, specific grid RNS path

```
grid fuse --mount --sandbox=grid:/home/vana/fuseDir local:/localdir/mountDir --daemon
```

To fuse mount the grid using uid, this optional attribute allows to use different user id for all file ownership mappings of the resources in GenesisII grid name-space When it is not specified, the current UNIX user id is used for all mappings by default.

```
grid fuse --mount --uid=50505 --sandbox=grid:/home/vana/fuseDir local:/localdir/mountDir --daemon
```

To Unmount the already fuse mounted directory

```
grid fuse --nmount local:/localdir/mountDir
```

ln

Similar to the ln tool in UNIX, the grid ln tool allows the user to make links to grid resources in the current session's RNS space. This can be done either from existing RNS paths, or from EPRs or degenerate EPRs (URLs). As with the UNIX ln tool, in the case where the link is made from an existing RNS path to a new one, the new path need not be given. In this case, the tool will assume that the user wishes to create a link from the source path to a similarly named entry in the user's current working RNS path.

It is important to note that unlike UNIX, RNS links behave neither like UNIX hard-links, nor like UNIX soft-links. In fact, their behavior lies somewhere between those two UNIX concepts. Links in the grid are hard-links in the sense that each link directly specifies an EPR to reference (just has hard links in UNIX directly indicate an inode). However, unlike hard-links in UNIX, grid links do not perform any reference counting, thus giving a behavior similar to UNIX soft-links whereby a resource can be removed or terminated leaving danging entries elsewhere in the name-space (or in other name-spaces).

To link one directory to another

```
grid ln lnDir1 lnDir2
```

To link one file to another

```
grid ln test1.txt test2.txt
```

Linking and already linked file

```
grid ln test2.txt test3.txt
```

To link an epr file to a resource. When the epr-file option is given, the user is identifying a local file-system file, rather than an RNS path, to use for the source of the link. Because RNS space links human readable names to EPRs, it is possible using this command option to link in a completely new (to the given RNS space) grid resource based off of that resource's EPR. The file indicated by this option should be a properly formatted XML document containing a WS-Addressing EndpointReferenceType element.

```
grid ln --epr-file=/home/vana/eprFile-test1 /home/vana/test1-epr-link
```

To link an epr file to a resource

```
grid ln --no-lookup -epr-file=/home/vana/eprFile-test1 /home/vana/test2-epr-link
```

When the service-url option is given, the user indicates the URL of a grid service that he or she wishes to link into the RNS space. This is done by forming the degenerate WS-Addressing EndpointReferenceType indicated by this URL (i.e., by creating an EPR whose address field is the given URL, and for whom all other fields are empty or null).

```
grid ln --no-lookup --service-url=local:./testServiceUrl.txt /home/vana/serviceUrl-link
```

To link a Container manually to an existing grid

```
grid ln --service-url=https://dnsname:port/axis/services/VCGRContainerPortType
/home/vana/newContainer
```

unlink

The unlink tool is a simple grid tool which allows a user to unlink an entry from RNS space. This essentially means that the RNS mapping from that name to the indicated resource will be removed. Because links are not reference counted in the grid, this tool has no effect on the resource itself. In other words, unlink does not destroy target resources, it merely removes there names from RNS space. To destroy a target resource, use the rm command. When a local path is provided, this tool works the same as a regular removal.

```
grid unlink   /home/vana/lnDir2 (lnDir2 is linked using above 'ln' command)
```

ls

Similar to the ls tool in UNIX, the grid ls tool allows the user to list entries in both local and RNS directories.

To List a file in grid name space

```
grid ls /home/vana/testFile.txt
```

To long list the directories only

```
grid ls -ld /home/vana/
```

To get EPR listing of the file

```
grid ls -e /home/vana/testFile.txt
```

To long list all the directories with EPR

```
grid ls -led /home/vana
```

To List a directory contents of a local directory

```
grid ls -le local:/localdir
```

mkdir
Similar to the mkdir tool in UNIX, the grid mkdir tool allows the user to create a directory relative to the current directory (or the absolute path). To indicate a local path, make sure to begin the pathname with 'local:'.

To create a directory

```
grid mkdir /home/vana/newDir
```

To recursively create directories

```
grid mkdir -p /home/vana/recDir/newDir/recDir
```

To create a directory on a specific container, gives the path of an RNS service to use for creation. User should have permission to create directories on that container

```
grid mkdir --rns-service=${containerPath}/Services/EnhancedRNSPortType ${rnsPath}/RNSDir
```

mv
Similar to mv in the UNIX operating system, the grid mv command moves(renames) a resource. The source and the target must both be local paths, or they must both be grid paths. Moving from local to grid name-space or from grid to the local name-space is not permitted. During grid path to grid path move, the target path should not already exist. The renamed resource is in the same-container as the "source" resource. In, a local path to local path move, an already existing target-path causes a replacement of the existing file.

To move one file to another (rename)

```
grid mv /home/vana/oldFile.txt /home/vana/newFile.txt
```

To move a directory to another directory

```
grid mv /home/vana/oldDir /home/vana/newDir
```

To move local directory to another local directory

```
grid move local:/localdir/oldDir local:/locadir/newDir
```

rm
Similar to the rm tool in UNIX, the grid rm tool allows the user to delete/remove resources on the Grid or local file-system relative to the current path or an absolute path. To indicate a local path, make sure to begin the pathname with 'local:'.

To remove a file from grid name-space

```
grid rm /home/vana/testFile.txt
```

To remove a directory from grid name-space

```
grid rm /home/vana/testDir
```

To remove a file from local directory

```
grid rm local:/localdir/testDir
```

To remove multiple files in grid name-space
```
grid rm /home/vana/testFile1.txt /home/vana/testFile2.txt
```
OR
```
grid rm /home/vana/testFile*
```

5. General Tools

client-ui
Launches the GUI version of the GenesisII client.

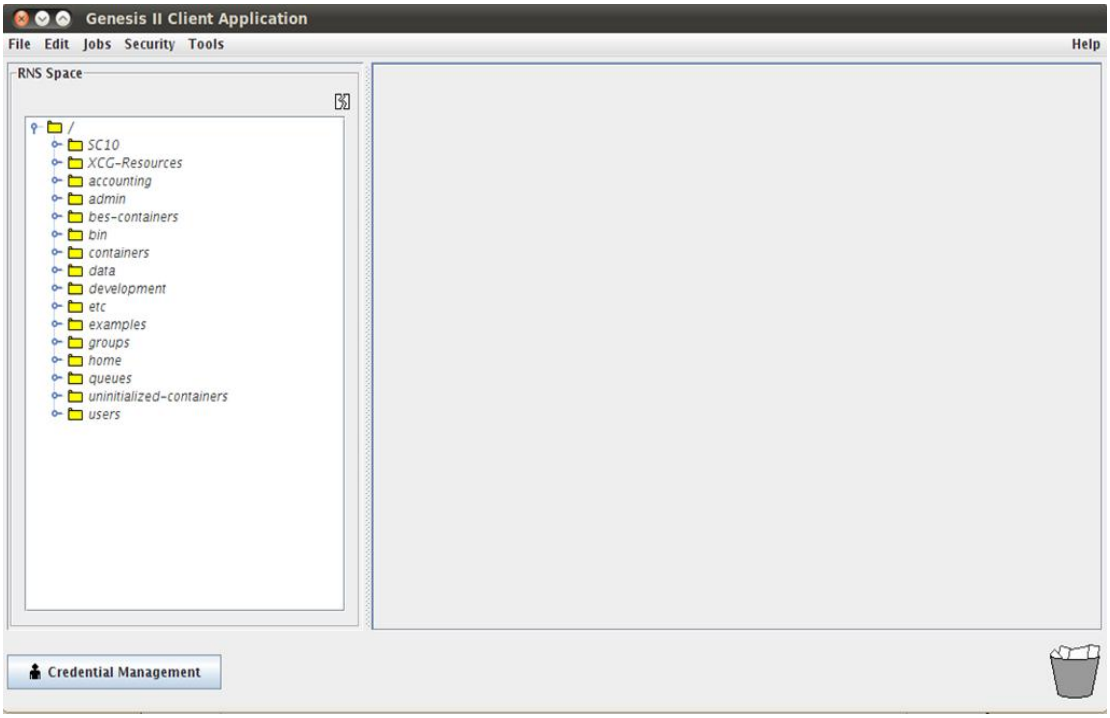

Figure. 4 GUI launched using clinet-ui command

history
Display the list of up to five-hundred recently issued commands with line numbers.

To list all the top-500 history events

```
grid history
```

To list last 3 history commands

```
grid history 3
```

To clear all history commands

```
grid history -c
```

ping
The ping diagnostic tool sends a query-string to the specified resource. If the user is authorized and the resource is available, the resource will echo the query-string back to the user. The round-trip exchange is attempted the specified number of times.

To ping a directory

```
grid ping /home/vana
```

To ping a BES resource's

```
grid ping /home/vana/myBES
```

pwd
Similar to the pwd tool in UNIX, the grid pwd tool allows the user to discover his/her working directory. There are no options for pwd

schedule-termination
Sets a schedule for when the indicated resources will automatically be terminated from the server.

To schedule termination of a directory after 10 minutes. Calendar value is a string which indicates the time at which the users wishes for a resource to be terminated automatically. This string can either be a valid Java parseable date string, or a + sign followed by a valid duration strings (+number[ms|s|m|h|d]). It is highly recommended that users use the duration strings version of this tool as the date strings that the native Java parser understands.

```
gridschedule-termination /home/vana/tempDir +10m
```

Notes:
Currently, the Genesis II system performs lazy deletion of resources scheduled to terminate. This means that the resources can expire without actually being garbage collected. Resources terminated on a schedule are not garbage collected until an attempt is made to access them (after their termination time).

script
The script command allows a user to run an Xscript program on the grid. The XML scripting language is a very simple scripting language which allows users to issue any Genesis II command. It also provides mechanisms for simple control flow as well as simple macro management. This tool is available to all users, but intended primarily for Genesis II developers. JavaScript scripts can also be executed. Filename extensions are used to determine the correct language to use when running scripts. Thus, to run a JavaScript script, one would indicate a file whose filename ended in the .js extension. Similarly, to run an XScript script file, the filename must end with the .xml filename extension.

To run script commnad, you should have a Xscript (xml based script) file with grid or shell commands in it.

```
grid script local:/locadir/myScript.xml
```

You can also pass parameters to your XScript

```
grid script local:/locadir/lsScript.xml /home/vana
```

`lsScript.xml` : Xscript to list contents of any given directory

```
<?xml version="1.0" encoding="UTF-8"?>
<gsh:script xmlns:gsh="http://vcgr.cs.virginia.edu/genii/xsh/script"
xmlns:geniix="http://vcgr.cs.virginia.edu/genii/xsh/grid">

        <gsh:define name="Path" source="${ARGV[1]}"/>
        <geniix:ls>
                <gsh:param>-l</gsh:param>
```

```
            <gsh:param>${Path}</gsh:param>
        </geniix:ls>
</gsh:script>
```

To learn more about Xscript refer to
http://genesis2.virginia.edu/wiki/Main/XScriptLanguageReference

set
Used to add or change a grid user-environment variable. If the environment variable does not exist, then it is set. If the environment variable already exists, then it is set to the new value. The environment variable can be used by wrapping ${} around the variable. Multiple environment variables can be used in the same command.

```
grid set myFile=/home/vana/set-file.txt
```

You can use this ${myFile} in consecutive commands

```
grid cat ${myFile}    // Should list the contents of set-file.txt
```

set-resource-properties
This command is used to set resource-properties of a given target-resource based on the given file. The resource-properties can be viewed using get-attributes command.

To set resource properties of a directory, first generate a resource properties file using get-attributes command and saving the output to a file. Use this resource properties file to set properties of another resource (in this case directory).

```
grid set-resource-properties /home/vana/newDir /home/vana/resourcePropertiesFile
```

set-user-config
Updates the local GenesisII installation to use specified deployment directory.

To set user config directory to “/lcoaldir/genii-installation-dir”
```
grid set-user-config local:/localdir/genii-installtion-dir
```

Note: By default GenesisII client or container chooses ~/.genesisII-2.0 as the deployment directory and client uses this directory as its state directory. If you want to change it to some other directory use the above command.

update
This functionality is not supported through the normal command line client. Instead, if you want to manually update the grid client, you need to exit this grid client (and any other grid software that you might have running on your local machine) and instead issue the separate "grid-update" program.

user-preferences
This command is used to modify and view the current user-preference. The options include to change the shell-prompt name, set the exception handler level to Simple/Debug, and also set the preferred mode to GUI or a shell-mode. The Debug exception handler provides a detailed stack-trace for the exception, while the Simple Exception Handler provides a brief description of the exception.

To change user preferences

```
grid user-preferences
```

This will invoke GUI, user preference options are self explanatory and can be changed using the following GUI.

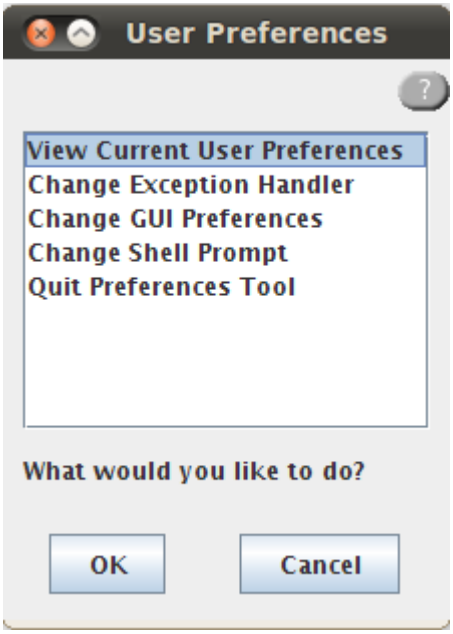

Figure. 5 GUI to change user preferences

6. Internal Use
These commands are mostly used by grid administrators only.

GetUserDir
Returns the path to the GenesisII state directory. This path is in the local file-system namespace.

To see what the current user directory is set to

grid getUserDir

cloudTool
This tool is used for interacting with a cloudBES resource. The options can be used for spawning VMs, shrinking VMs, killing a particular VM instance, querying the status of the Vms.

Command Options:
cloudTool [--shrink=<count>] <path-to-cloud-bes>
cloudTool [--spawn=<count>] <path-to-cloud-bes>
cloudTool <path-to-cloud-bes>
cloudTool [--kill=<id>] <path-to-cloud-bes>
cloudTool [--vmstatus] <path-to-cloud-bes>

container-stats

Displays the usage statistics of a particular container. The statistics displayed include the container start-time, database-access statistics and method-access statistics. Both the method-access statistics and the database-access statistics are categorized into three time-frames - statistics for the last 30 seconds, 1 minute and 5 minutes. The database statistics give the following information : Number of database connections opened, closed and the average time-duration of a connection for each time-frame. The method statistics give information about the total number of calls started, succeeded, failed and the average duration of a call. Additionally, there is also a class-wise break up of the total method-information.

To get stats on a container, user should have access permissions on that container

```
grid container-stats /containers/BootstrapContainer

Partial ample output:
Container start time:  Mon Dec 05 12:02:01 EST 2011
Database Statistics:
  Thirty Seconds:
    Num Opened = 56, Num Closed = 55, Average Duration = 5 ms
  One Minute:
    Num Opened = 87, Num Closed = 86, Average Duration = 3 ms
  Five Minutes:
    Num Opened = 336, Num Closed = 335, Average Duration = 1 ms
Method Statistics:
  Thirty Seconds:
    Totals:  Calls Started = 4, Calls Succeeded = 3, Calls Failed = 0, Failure Rate = 0.00%, Average Duration = 11 ms
    Class Totals(edu.virginia.vcgr.genii.container.rns.EnhancedRNSServiceImpl):  Calls Started = 3, Calls Succeeded = 3, Calls Failed = 0, Failure Rate = 0.00%, Average Duration = 11 ms
      Method (lookup):  Calls Started = 3, Calls Succeeded = 3, Calls Failed = 0, Failure Rate = 0.00%, Average Duration = 11 ms
```

idp

IDP – Identity provider. This command is usually used to add users to group(s). This is usually used by grid administrator to add new or existing user to group(s)

```
grid idp <container-path>/Services/X509AuthnPortType /groups/group-name
```

7. Job/Execution

Using these commands user will be able to define and run jobs on the grid. Grid administration usually would have created “Queue” and attach “BESes' to the queue. Users usually submit jobs to the “Queue” or directly to the “BES”.

job-tool

Launches Grid Job Tool which is used to create JSDL files. If a grid project-file list is provided then those files in the list which already exist are opened, while those that do not exist are created.

`grid job-tool` OR `gridjob-tool myJob`

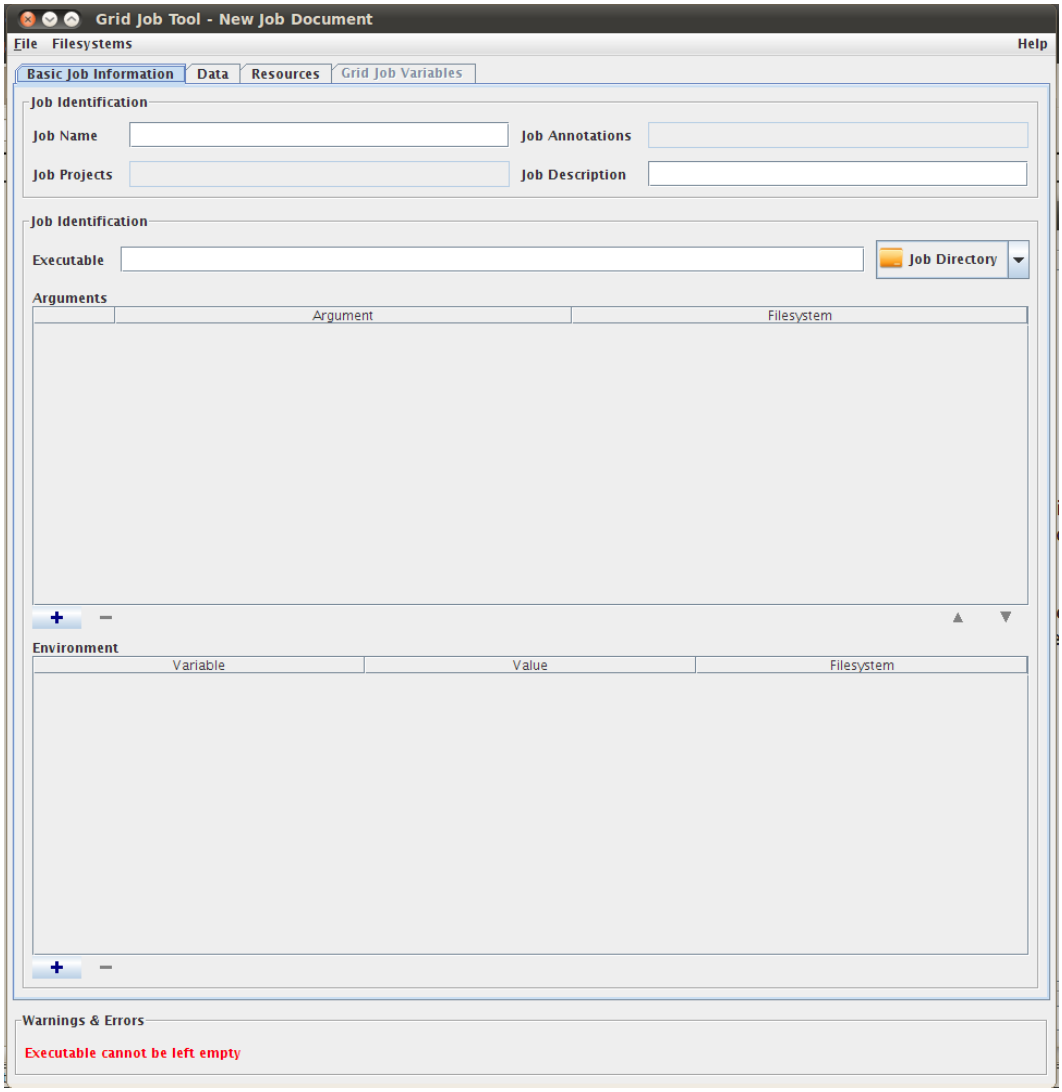

Figure. 6 job-tool GUI

For more information on how use job-tool, please refer
http://genesisii.cs.virginia.edu/docs/running-a-job-v1.0.pdf

parseJSDL
This command takes in a JSDL format job-file and outputs a serialized (binary) version of it.

To generate a binary file of your job (jsdl file)

```
grid parseJSDL /home/vana/ls.jsdl /home/vana/ls.binary
```

Sample ls.jsdl file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl"
xmlns:ns2="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
xmlns:ns3="http://schemas.ggf.org/jsdl/2006/07/jsdl-hpcpa"
xmlns:ns4="http://schemas.ogf.org/jsdl/2007/02/jsdl-spmd"
xmlns:ns5="http://vcgr.cs.virginia.edu/jsdl/genii" xmlns:ns6="http://docs.oasis-
```

```
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:ns7="http://schemas.ogf.org/hpcp/2007/11/ac"
xmlns:ns8="http://schemas.ogf.org/jsdl/2009/03/sweep"
xmlns:ns9="http://schemas.ogf.org/jsdl/2009/03/sweep/functions">
       <JobDescription>
              <JobIdentification>
                     <JobName>Ls Test</JobName>
              </JobIdentification>
              <Application>
                     <ns2:POSIXApplication>
                            <ns2:Executable>/bin/ls</ns2:Executable>
              </ns2:POSIXApplication>
              </Application>
       </JobDescription>
</JobDefinition>
```

qcomplete

The queue manages all jobs that are submitted to it from the time that they are submitted until the time that the users completes them. Even jobs in a final state such as FINISHED, CANCELLED, or FAILED are managed by the queue until they are completed. Completing a job simply means that the users wishes to garbage collect, or remove the job, from the queue.

To remove all FINISHED/CANCELLED/FAILED jobs from grid queue

```
grid qcomplete --all
```

To removed specific jobs from the queue

```
grid qcomplete <queue-path> ticket#
```

Note: Ticket# is returned when you submit a job to the queue using qsub command.

qconfigure

This command is usually used by user who is setting up a BES container or grid administrator. Users indicate resources that the queue can schedule jobs on by linking those resources (BES containers) into the queue's RNS space. Similarly, users can always see the list of existing resources in the queue by performing a grid ls inside the queue's namespace. The configure command allows the user to alter the number of slots available on a specific resource inside the queue, effectively limiting the number of simultaneous jobs that the queue can allocate to that resource. The default slot count for a resource (when first added to the queue) is 1.

To set the slot count to set on a BES

```
grid qconfigure <queue-path> <besName> 10
```

qkill

The qkill command allows grid users to terminate any managed job (not already in a final state) that they previously submitted. The qkill command will fail if any of the jobs indicated by the job-ticket list is already in a final state (hence, a slight race condition occurs in that a job may move from a non-final state to a final one between the user issuing the command and the command actually reaching the remote queue). Further, the qkill command will fail if any of the tickets specified are not owned by the caller.

To kill one job

```
grid qkill <queue-path> ticket#
```

To kill N jobs

```
grid qkill <queue-path> Ticket#1 Ticket#2 Ticket#3 …... Ticket#N
```

qlist

The qlist command lists all of the jobs currently managed by a given queue. This command returns a reduced set of information (information assumed to be publicly available) for all jobs currently in the queue. For a more detailed list of information, users should use the qstat command which only works with jobs that the caller owns, but returns both public, and private information.

To list all the jobs in a particular queue

```
grid qlist <queue-path>
```

qquery

The qquery command is used to obtain error information associated with a particular job in a particular queue.

To obtain information of a particular job

```
grid qquery <queue-path> ticket#
```

qreschedule

This command is used to return an already-running job back to the queue and ensures it is not rescheduled on same bes. WARNING: The slot count for this resource must be manually reset later. This command is useful when the Queue consists of BES' which interface to a queuing system like PBS. So, a job may be in the Running state on the grid, but on a Queued state on the back-end PBS. Such jobs, can be moved to an alternate BES where it can be executed immediately.

To reschedule one job

```
grid qreschedule <queue-path> ticket#
```

To reschedule N jobs

```
grid qreschedule <queue-path> ticket#1 ticket#2 ticket#3 …... ticket#N
```

qstat

The qstat command gives the status information for jobs currently managed by the queue and owned by the caller. If no tickets are provided, the qstat tool will stat all jobs owned by the current caller. If any tickets are given, only the jobs specified will be stat'ed. If the caller attempts to stat a job which he or she does not own, then the qstat call will fail. This tool is similar to qlist except that it only stats jobs owned by the caller and returns both public and private information about the jobs.

To get stats on single jobs currently in queue

```
grid qstat <queue-path> ticket#
```

To get information on N jobs in the queue

```
grid qstat <queue-path> ticket#1 ticket#2 ….... ticket#N
```

To get verbose information on all the job in queue owned by user

```
grid qstat --full <queue-path>
```

qsub
The qsub command is used when a user wishes to submit or add a new job to a queue for that queue to manage.

To submit a job to the queue, you should have a job description file (ex. ls.jsdl)

```
grid qsub <queue-path> local:/localdir/ls.jsdl
```

To Assign priority to your job. This option allows the user to specify a custom priority for the jobs that he or she is submitting. Priorities are integers in the range [-10, 10] and for which a lower number indicates a higher priority (i.e., that the jobs will run earlier then lower priority jobs).

```
grid qsub <queue-path> --priority=10 local:/localdir/ls.jsdl
```

Note: qsub returns job ticket# after successfully submitting the job. This ticket# can be later used to query the job, kill the job etc.

resource-history
This command is used to display the job-history of a particular job. The job-history is displayed as a series of chronological events. Events are categorized as Default, CreatingJob, CreatingActivity, Scheduling, StageIn, StageOut, ReQueing, Checking, Cleanup, Terminating, CloudSetup. The job-history can be serialized and written to a file or displayed on the console without serialization. This tool has a minimum event-level of "Trace". The available event-levels are Trace, Debug, Information, Warning and Error. To set a minimum event-level while viewing job-history, the GUI QueueManager tool needs to be used.

To get the job resource history of a job

```
grid resource-history --dump=local:/localdir/resourcefile.txt <queue-path> ticket#
```

run
This command will run a specified JSDL file on a target BES container, or through a scheduler. It can be used to start jobs, and to check on their status. It can also run executables, and allows arguments to be passed to those executables.

To run a job asynchronously, specify a RNS path where job information will be stored. You can later check this path for status of the jobs. When asynch option is used the shell will return the prompt immediately.

grid run --name=LS-Test --async-name=/home/vana/ls-job /home/vana/myBES --jsdl=local:/localdir/ls.jsdl

To check the status of async jobs

```
grid run --check-status /home/vana/ls-job
```

To run a job synchronously, when asynch option is NOT used the shell will NOT return the prompt until job is in FINISHED/CANCELLED/ERROR state.

```
grid run --name=LS-Test /home/vnaa/myBES --jsdl=local:/localdir/ls.jsdl
```

runJSDL
This command is used to execute a job "locally" - i.e in the client-machine. The job is described in terms of a job-file which is either in JSDL or binary format. The staging operations occur into (and out of) the working directory.

To run a job using jsdl

```
grid run ./localDir local:/localDir/ls.jsdl --type=jsdl
```

To run a job using binary file (binary file can be generated using parseJSDL option)

```
grid run ./localDir local:/localDir/ls.binary --type=binary
```

8. Misc
These are more generic commands which probably dint fit into any other category :)

connect
The connect tool allows a user to manually connect to a given remote net. Normally users shouldn't have to use this tool. By specifying the location of an input stream which contains information about the root RNS of a namespace, the client essentially identifies the exact root that he or she wishes to communicate with. The optional deployment directory parameter allows the user to further specify configuration information necessary to communicate with that net. By default, the deployment directory chosen will be deployments/default.

To connect to an already existing grid Eg. Current XCG grid

```
grid connect http://vcgr.cs.virginia.edu/XCG/2.0/root.xml GeniiNetFullContainer
```

create-resource
The create-resource command, creates a new resource using generic creation mechanisms.

To create a BES resource

```
grid create-resource /home/vana/newContainer/Services/GeniiBESPortType /home/vana/newBES
```

To create a RNS resource (newDirectory physically resides on the disk space on newContainer)

```
grid create-resource /home/vana/newContainer/Services/EnhancedRNSPortType home/vana/newDirectory
```

echo
Similar to echo in the UNIX operating system, the grid echo command outputs String arguments to the terminal. Extra spaces may be enclosed in double quotes. Environment variables wrapped in ${} are automatically substituted.

To echo strings to terminal

```
grid echo test-string test-string test-string
```

To echo something to a file

```
grid echo test-string > /home/vana/output.txt
```

tty
TTY Objects can received output from various sources and forward that information on to watchers. The buffer that it uses to do this is limited in size and must be read relatively frequently or information will be lost. Any jobs the user submits to a BES container (directly or through a queue) will have their output forward back through the TTY object.

To be able to watch on for eg. BES, do the following

```
grid create-resource /containers/BootstrapContainers/Services/TTYPortType /home/vana/tty-object
grid tty watch /home/vana/tty-object
grid tty unwatch
```

9. Security
Most of the commands are used by grid administrators, although some of the commands are used by users as well.

IDPLogin
To add a user to a group they need to be in the execute acl of the sts (IDP). There are two ways they can login to it.

1. You can link the group into their login sts (i.e ln /group/groupname /users/userid/groupname). This automatically logs them in after they log in to their primary STS (Secure Token Service).

2. If we dont link them in, then they can manually login to it with the IDPLogin command, after they have logged into their primary successfully.

To login to a group called specialGroup using IDPLogin

```
grid login --username=user1 --password=pass1
grid IDPLogin /groups/specialGroup
```

authz
The interactive authz tool gives users the ability to inspect and modify the authorization policy for a grid resource. Once invoked, the authz tool retrieves the authorization policy of the indicated resource and displays it to the user. The user is then given interactive menu options for:

* Toggling the message-level-encryption requirement for the resource (disabled by default). Enabling this instructs the resource to only accept and respond with messages that have been encrypted as per SOAP XML Encryption (http://www.w3.org/TR/xmlenc-core/).
* Modifying portions of the retrieved policy.
* Committing the (updated) policy back to the resource. Modifications to policy will not take effect until the updated policy has been committed to the resource.

Authorization policies for GenesisII resources are represented as access control lists (ACLs). Operations upon a grid resource fall into one of three categories: Read, Write, and Execute (R/W/X). GenesisII resources manage three separate "allow" ACLs, one for each R/W/X category. The authz tool allows users to inspect, add, and remove identities from these R/W/X ACLs using the chmod tool syntax.

To inspect or modify authorization policy of a grid resource

```
grid authz /home/vana
```

chmod

The chmod tool gives users the ability to modify the authorization policy for a GenesisII grid resource. Authorization policies for GenesisII resources are represented as access control lists (ACLs). Operations upon a grid resource fall into one of three categories: Read, Write, and Execute (r/w/x). GenesisII resources manage three separate "allow" ACLs, one for each r/w/x category. The authz tool allows users to add and remove identities from these R/W/X ACLs.

create-user

This command is used to create new users in the grid namespace.

To create new user

```
grid create-user /containers/BootstrapContainer/Services/X509AuthnPortType user1 --login-name=user1
--login-password=password1
```

Just the above step will create a user in `/containers/BootstrapContainer/Services/X509AuthnPortType`
To create user1' s home and link to certain group, following steps are needed

```
grid ln /containers/BootstrapContainer/Services/X509AuthnPortType/user1 /users/user1
grid ln /groups/SomeGroup /users/user1/SomeGroup
grid chmod /groups/SomeGroup +x /users/user1
grid mkdir /home/user1
grid chmod /home/user1 +rwx /users/user1
```

create-user-delegate

This command is used to delegate all the ACL's of an existing user to new user. Once succesfull the new user should have same ACL on the grid resources as previous user.

To create and delegate Acls to new user

```
grid create-user-delegate --storetype=PKCS12
/containers/BootstrapContainer/Services/X509AuthnPortType
$/containers/BootstrapContainer/Services/X509AuthnPortType/newUser --login-name=$newUser --login-
password=newPassword
```

keystoreLogin

```
grid keystoreLogin --no-gui --password=somepass --pattern=somepattern --storetype=PKCS12
local:$GENII_INSTALL_DIR/deployments/default/security/someFile.pfx
```

login

The login tool allows users of the Genesis II system to acquire a set of credentials that will the be used for authentication and authorization purposes. This tool is very flexible in terms of the types and sources of credentials that it can use. In the case where login has no parameters at all, the tool assumes a set of reasonable defaults and prompts the user for only a minimal set of values with which to log in. For most users, this version of the tool should suffice.

To Login to grid

```
grid login --username=user1 --password=password1
```

If you do not give username or password in the command line, you will be prompted to provide on

```
grid login
```

Figure.7 username prompt

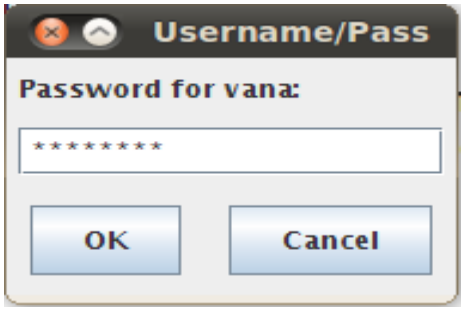

Figure.7 Password prompt

User can login any number of times with same username without ever logging out but it is not

recommended. If user has multiple logins and he can login with multiple logins without logging out again not recommended for security reasons (username with higher privileges can accidentally do something unwanted if logged in as multiple users).

Logout
The logout command is used to remove authentication information from the user's context.

To logout completely of grid (this will logout of all the login's I.e if user logged in with multiple logins)
```
grid logout
```

To logout from a specific username or pattern (pattern can be username, password, DN name etc)
```
grid logout --pattern=passWord1
grid logout --pattern=user1
```

passwordLogin
This command is used to create a username-token which is added to the calling context. More than one username-token credential cannot be possessed. The whoami command can be used to view the list of credentials.

To login using passwordLogin
```
grid passwordLogin --username=user1 --password=password1
```

shell-login
This tool allows a user to "login" to a particular directory, and then executes the .glogin.xml or .glogin.js script here. If both .glogin.xml and .glogin.js exist, then both scripts are executed.

For this you will need a file called .glogin.xml, this file should be in a grid directory on which user have access permissions, suppose /home/user1/.glogin.xml
```
<?xml version="1.0" encoding="UTF-8"?>

<gsh:script xmlns:gsh="http://vcgr.cs.virginia.edu/genii/xsh/script"
xmlns:geniix="http://vcgr.cs.virginia.edu/genii/xsh/grid">

<geniix:login>
       <gsh:param>--username=user1</gsh:param>
       <gsh:param>--password=password1</gsh:param>
</geniix:login>

</gsh:script>
```

To login using shell-login
```
grid shell-login /home/user1
```

whoami
The command whoami prints out the credentials of the currently logged in user.

```
grid whoami
```