Installation

What is HUBzero?

HUBzero is a platform used to create dynamic web sites for scientific research and educational activities. With HUBzero, you can easily publish your research software and related educational materials on the web. Powerful middleware serves up interactive simulation and modeling tools via your web browser. These tools can connect you with rendering farms and powerful Grid computing resources.

Minimum System Requirements

HUBzero installations require one or more dedicated physical hosts running Debian GNU/Linux 6.0. In particular there are version specific requirements on mysql (5.1), python (2.6) and php (5.3).

Other debian based distributions might theoretically work with some modification, although they would be totally unsupported.

A typical starter HUBzero installation might consist of a single physical server with dual 64-bit quad-core CPUs, 16 Gigabytes of RAM and a terabyte of disk.

It is possible to run HUBzero inside of a virtual machine such as ones created by VMware and VirtualBox. While fully functional there will be significant performance and resource limitations in such an environment. For demonstration or development purposes we often create VM images with less than a gigabyte of RAM and 5 gigabytes of disk. While fully functional, these virtual machines would only be suitable for a single user doing development or testing.

Target Audience

This document and the installation and maintenance of a HUBzero system has a target audience of experienced Linux administrators (preferably experienced with Debian GNU/Linux).

How To Get Source Code

This is optional

You'll find most of the source code within the web root of a working hub. But you can get source code for the middleware and all other parts of HUBzero by retrieving the source code packages using the debian package manager. To do so, first, update your package mechanism to pull packages from our HUBzero repository (codename 'manny' for this release). Edit your
/etc/apt/sources.list and add the following line::

deb-src http://packages.hubzero.org/deb shira main contrib non-free

Then, install the hubzero archive key so that your system can verify packages from the hubzero archive. Run the following command as root:

wget http://packages.hubzero.org/deb/hubzero-signing-key.asc -q -O - | apt-key add -

Once the public key for http://packages.hubzero.org has been installed, you can request the hubzero source code for each package:

apt-get source packageName

where **packageName** is the desired package from the table below.

## Package List

The following packages (subject to change): are available from the HUBzero repository for this release. Not all are necessarily used for any given installation.

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>hubzero-app</td>
<td>Command line tool to manage installation/publishing of HUBzero apps</td>
</tr>
<tr>
<td>hubzero-app-workspace</td>
<td>HUBzero App providing a lightweight Linux desktop, for app/tool development</td>
</tr>
<tr>
<td>hubzero-chuse</td>
<td>HUBzero graphical front end to the user environment manager 'use'</td>
</tr>
<tr>
<td>hubzero-cli</td>
<td>HUBzero command line interface tools</td>
</tr>
<tr>
<td>hubzero-cms-1.2</td>
<td>The HUBzero Content Management System (based on Joomla! framework)</td>
</tr>
<tr>
<td>hubzero-expire-sessions</td>
<td>Expires unused app/tool sessions</td>
</tr>
<tr>
<td>hubzero-filexfer</td>
<td>Transfer files between App Sessions and user's desktop</td>
</tr>
<tr>
<td>hubzero-filexfer-xlate</td>
<td>Support daemon for the filexfer program</td>
</tr>
<tr>
<td>Package Name</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>hubzero-firewall</td>
<td>HUBzero firewall that protects app/tool sessions</td>
</tr>
<tr>
<td>hubzero-forge</td>
<td>Creates project areas for tool development</td>
</tr>
<tr>
<td>hubzero-icewm</td>
<td>Linux ICE window manager configuration, used in workspaces</td>
</tr>
<tr>
<td>hubzero-icewm-captive</td>
<td>Linux ICE window manager specially crafted to support tools in a session</td>
</tr>
<tr>
<td>hubzero-icewm-themes</td>
<td>The HUBzero Linux ICE window manager theme, used in workspaces</td>
</tr>
<tr>
<td>hubzero-invokeapp</td>
<td>HUBzero application invoke script</td>
</tr>
<tr>
<td>hubzero-mailgateway</td>
<td>HUBzero mail gateway</td>
</tr>
<tr>
<td>hubzero-metrics</td>
<td>HUBzero metrics support package</td>
</tr>
<tr>
<td>hubzero-mw-client</td>
<td>HUBzero middleware - client</td>
</tr>
<tr>
<td>hubzero-mw-service</td>
<td>HUBzero middleware - execution host session manager</td>
</tr>
<tr>
<td>hubzero-mw-session</td>
<td>HUBzero middleware - per session tools</td>
</tr>
<tr>
<td>hubzero-mysql</td>
<td>MySQL configuration package for HUBzero</td>
</tr>
<tr>
<td>hubzero-openldap</td>
<td>OpenLDAP configuration package for HUBzero</td>
</tr>
<tr>
<td>hubzero-openvz</td>
<td>OpenVZ configuration package for HUBzero</td>
</tr>
<tr>
<td>hubzero-policyrcd</td>
<td>HUBzero policy-rc.d for invoke-rc.d</td>
</tr>
<tr>
<td>hubzero-python</td>
<td>HUBzero python API module</td>
</tr>
<tr>
<td>hubzero-rappture</td>
<td>The Rapid APPlication infrastrucTURE toolkit for building scientific tools</td>
</tr>
<tr>
<td>hubzero-rappture-session</td>
<td>Session support packages for Rappture</td>
</tr>
<tr>
<td>hubzero-ratpoison-captive</td>
<td>Linux window manager, used in app/tool sessions</td>
</tr>
<tr>
<td>hubzero-submit-client</td>
<td>HUBzero submit client</td>
</tr>
<tr>
<td>hubzero-submit-common</td>
<td>HUBzero submit common python library</td>
</tr>
<tr>
<td>hubzero-submit-condor</td>
<td>HUBzero condor build for hubzero-submit</td>
</tr>
<tr>
<td>hubzero-submit-distributor</td>
<td>HUBzero middleware submit distributor</td>
</tr>
<tr>
<td>hubzero-submit-monitors</td>
<td>HUBzero middleware submit monitors</td>
</tr>
<tr>
<td>hubzero-submit-pegasus</td>
<td>HUBzero pegasus build for hubzero-submit</td>
</tr>
<tr>
<td>hubzero-submit-server</td>
<td>HUBzero job submission server</td>
</tr>
<tr>
<td>hubzero-subversion</td>
<td>HUBzero Subversion support package</td>
</tr>
<tr>
<td>hubzero-telequotad</td>
<td>Disk quota monitor</td>
</tr>
<tr>
<td>hubzero-textifier</td>
<td>HUBzero textifier</td>
</tr>
<tr>
<td>hubzero-texvc</td>
<td>Helper utility to generate math formulas for wiki markup</td>
</tr>
<tr>
<td>hubzero-trac</td>
<td>HUBzero Trac support package</td>
</tr>
<tr>
<td>hubzero-trac-mysqlauthz</td>
<td>Plug-in for MySQL user auth in project development areas</td>
</tr>
<tr>
<td>hubzero-twm-captive</td>
<td>Linux TWM window manager, used in app/tool sessions</td>
</tr>
<tr>
<td>hubzero-use</td>
<td>Command for configuring the environment within a workspace</td>
</tr>
<tr>
<td>hubzero-use-apps</td>
<td>HUBzero apps environment for 'use'</td>
</tr>
<tr>
<td>Package Name</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>hubzero-usermap</td>
<td>File permission mapping FUSE filesystem used by WebDAV</td>
</tr>
<tr>
<td>hubzero-vncproxy</td>
<td>HUBzero VNC proxy helper</td>
</tr>
<tr>
<td>hubzero-vncproxyd</td>
<td>HUBzero VNC proxyserver</td>
</tr>
<tr>
<td>hubzero-webdav</td>
<td>HUBzero WebDAV support package</td>
</tr>
<tr>
<td>libapache2-vncproxy</td>
<td>HUBzero vncproxy module for apache2-mpm-prefork</td>
</tr>
<tr>
<td>php-oauth</td>
<td>OAuth PECL Library for PHP 5</td>
</tr>
<tr>
<td>php-stem</td>
<td>Stemming PECL Library for PHP5</td>
</tr>
<tr>
<td>tightvnc-java</td>
<td>Modified VNC Client that receives app/tool sessions within a web browser</td>
</tr>
<tr>
<td>vnc4server</td>
<td>Modified VNC Server that sends app/tool sessions to the web browser</td>
</tr>
</tbody>
</table>
Linux

Install Basic Operating System

The latest version of Debian GNU/Linux 6.0 (6.0.7 as of this writing) should be installed on each physical host used by a HUBzero installation.

Packaging of this release has been done with Debian 7.x in mind, but due to time constraints support for Debian 7.x is not yet available. When available it will still have significant limitations due to the fact that Debian no longer supports an OpenVZ kernel. HUBzero will be switching primary support to Redhat Enterprise Linux 6.x packages over the next year.

HUBzero has packaging support for amd64 (64bit) Intel architectures. i386 (32bit) packaging is possible but was not produced for this release due to lack of demand.

To install Debian GNU/Linux, you can easily obtain a copy, and then follow the installation instructions for your architecture.

Installing Debian GNU/Linux using a small bootable CD is the simplest method.

When the installation is complete your system will reboot into a Debian GNU/Linux system.

Don't forget to remove your installation media and/or change your server's boot media order if you changed them prior to installation.

The precise configuration (such as disk configuration, networking, etc) is dependent on how the hub is to be used and what hardware is being used. These instructions outline the simplest "hub in a box" configuration but may not be suitable for larger sites. It is expected that the hub will be managed by an experienced Linux administrator who can help scale your site to the capacity required.

Set hostname

Throughout this documentation you will see specific instructions for running commands, with part of the text highlighted. The highlighted text should be modified to your local configuration choices. (e.g. replace "example.com" with the fully qualified hostname of your machine).

Optional. If you didn't specify the fully qualified domain name when running setup you will need to set it here.

HUBzero expects the `hostname` command to return the fully qualified hostname for the system.

# hostname example.com
To make the change permanent you must also edit the file /etc/hostname, this be done simply with:

```
# echo "example.com" > /etc/hostname
```

**Fix hosts**

Now edit /etc/hosts by making sure that a line exists that looks like

```
127.0.1.1   example.com   example
```

Any other lines with "127.0.1.1" should be removed.

**Delete local users**

HUBzero reserves all user ids from 1000 up for hub accounts. As part of the HUBzero middleware every account must map to a corresponding system account. Therefore when starting up a hub it is required to remove all accounts that have user ids 1000 or greater. On a new installation there is typically one such account that is created when you set up the hub, and this account can be removed as follows:

```
# rm -fr /home/username
# deluser username
```

If you require additional system accounts, they can be numbered between 500-999 without interfering with hub operations.

**Configure Networking**

*Optional.* If you didn't configure networking during installation you will need to do so now.

For help with networking setup try this [link](#).

**Setting up your IP address.**
The IP addresses associated with any network cards you might have are read from the file /etc/network/interfaces. This file has documentation you can read with:

```
# man interfaces
```

A sample entry for a machine with a static address would look something like this:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
  address 192.168.1.90
  gateway 192.168.1.1
  netmask 255.255.255.0
  network 192.168.1.0
  broadcast 192.168.1.255
```

Here we've setup the IP addresses, the default gateway, and the netmask.

For a machine running DHCP the setup would look much simpler:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface - use DHCP to find our address
auto eth0
iface eth0 inet dhcp
```

(If you're using a DHCP based setup you must have a DHCP client package installed - usually one of pump or dhcp-client.)

If you make changes to this file you can cause them to take effect by running:
# /etc/init.d/networking restart

## Setting up DNS

Use whatever nameserver and other options as recommended by your ISP. If you used DHCP to set up networking it is likely this has already been set.

When it comes to DNS setup Debian doesn't differ from other distributions. To cause your machine to consult with a particular server for name lookups you simply add their addresses to /etc/resolv.conf.

For example a machine which should perform lookups from the DNS server at IP address 192.168.1.10 would have a resolv.conf file looking like this:

```
nameserver 192.168.1.10
```

## Configure Advanced Package Tool

Now configure the location of the HUBzero package repository by adding the following line to /etc/apt/sources.list

```
debug http://packages.hubzero.org/deb shira main
```

You will need to get and install the hubzero archive key to be able to verify packages from the hubzero archive:

```
# wget http://packages.hubzero.org/deb/hubzero-signing-key.asc -q -O - | apt-key add -
```

Once the public key for http://packages.hubzero.org has been install you can then upgrade the current packages to their latest releases.

Look through the rest of the file. If you see references to "stable" you should change them to "squeeze" otherwise the system will try to upgrade you to Debian 7.x when you do an update because the current "stable" release is Debian 7.x.
# apt-get update
# apt-get upgrade
MySQL

Install

# apt-get install -y hubzero-mysql

You will be prompted to enter a MySQL administrative password.

Configure

Default configuration works well for starters. But for optimal performance you will need a database administrator capable of tuning your database to your hardware configuration and site usage.
**Mail**

**Install**

We need to install exim4 to enable outgoing email

```
# apt-get install -y exim4
```

**Configure**

```
# dpkg-reconfigure exim4-config
```

Configure mail as appropriate for your site and IT infrastructure. We outline a sample standalone configuration below. The requirement is for php to be able to send mail (registration confirmation and other notices need to go out) and for exim4 to receive mail (for support ticket and forum email gateway functions to work).

This is just an example of a standalone mail configuration.

**General type of mail configuration**

internet site; mail is sent and received directly using SMTP

**Mail name**

enter the fully qualified domain name (FQDN) of the host (example.com)

**IP-addresses to listen on for incoming SMTP connections**

leave blank (listen for connections on all available network interfaces)

**Other destinations for which mail is accepted**

leave blank or (equivalently) with local hostname (all local domains will be treated identically)

**Domains to relay mail for**

leave blank

**Machines to relay mail for**
INSTALLATION

leave blank

Keep number of DNS-queries minimal (Dial-on-Demand)
No

Delivery method for local mail
mbox format in /var/mail/

Split configuration into small files?
Yes

Test

Use a real email address below so you can see if you get the email

# Mail -v someone@gmail.com
CMS

Install

# apt-get install -y hubzero-cms-1.2

Configure

# hzcms install example
# a2dissite default default-ssl
# a2ensite example example-ssl
# /etc/init.d/apache2 restart
OpenLDAP

Install HUBzero LDAP support

# apt-get install -y hubzero-openldap

You will be prompted to enter a LDAP administrative password.

Some packages will ask you to configure them when you run this step. I am pretty sure you can just accept the defaults because we rewrite the configuration files during the next step anyway.

Configuring nslcd: LDAP server URI:

keep the default or enter "ldap://localhost/"

Configuring nslcd: LDAP server search base:

keep the default

Configuring libnss-ldapd

Select only "group", "passwd", "shadow"

Configure OpenLDAP Database

# hzldap init
# hzcms configure ldap --enable
# hzldap syncusers

Test

# getent passwd

You should see an entry for user 'admin' toward the end of the list if everything is working correctly.
WebDAV

Install WebDAV

# apt-get install -y hubzero-webdav

Configure WebDAV

# hzcms configure webdav --enable

Test

# ls -l /webdav/home/admin
Total 0

Browse to your site’s https/webdav address (e.g. https://myhub/webdav). You should get
prompted for a username and password. Use the admin account. You should see an empty
directory listing and no error messages.

Now test using a WebDAV client.

# apt-get install cadaver
# cadaver https://localhost/webdav

You will be prompted to accept self signed certificate (if it is still installed) and then to enter your
username and password. Use the 'admin' account again to test. When you get the
"dav:/webdav/>" prompt just enter "ls" and it should show the test file.

Finally clean up test case

# apt-get purge cadaver

Troubleshooting
If the test doesn't work, check if the fuse kernel module is loaded

```
# lsmod | grep fuse
fuse   54176   0
```

If there is no output then try starting the kernel module manually

```
# modprobe fuse
```

Then try the test again
Subversion

Install

# apt-get install -y hubzero-subversion

Configure

# hzcms configure subversion --enable
Trac

Install

# apt-get install -y hubzero-trac

Configure

# hzcms configure trac --enable
Forge

Install

# apt-get install -y hubzero-forge

Configure

# hzcms configure forge --enable
OpenVZ

Install

HUBzero makes extensive use of OpenVZ containers so it is recommended to use the OpenVZ enabled kernel on all HUBzero servers.

# apt-get install hubzero-openvz

Configure

# hzcms configure openvz --enable

If configuration is successful it should prompt you to reboot the server to activate the new kernel.

# reboot

Test

# vzlist
Container(s) not found

Or it will list the containers currently running if you check this on a running hub. The salient point being that the command doesn't issue any kind of error message.
**Firewall**

**Install**

```bash
# apt-get install -y hubzero-firewall
```

HUBzero requires the use of iptables to route network connections between application sessions and the external network. The scripts controlling this can also be used to manage basic firewall operations for the site. If you use manage iptables with other tools you will have to make sure the rules in these scripts are maintained. `/etc/firewall_on` and `/etc/firewall_off` turn the HUBzero firewall on and off respectively. Scripts in `/etc/rc.X/` to `/etc/mw/firewall_on` causes the script to run at startup (these links were created for you). The firewall is enabled in all boot modes 0-6. The basic scripts installed here block all access to the host except for those ports required by HUBzero (http,https,http-alt,ldap,ssh.smtp,mysql,submit,etc).
**Maxwell Service**

**Install**

# apt-get install -y hubzero-mw-service

**Configure**

# mkvztemplate amd64 squeeze shira

# hzcms configure mw-service --enable

**Test**

# maxwell_service startvnc 1 800x600 24

Enter an 8 character password when prompted (e.g., "testtest")

This should result in a newly create OpenVZ session with an instance of a VNC server running inside of it. The output of the above command should look something like:

Reading passphrase:

testtest

================================ begin /etc/vz/conf/hub-

session-5.0-amd64.umount =============================

Removing /var/lib/vz/root/1 :root etc var tmp dev/shm dev

================================ end /etc/vz/conf/hub-

session-5.0-amd64.umount =============================

stunnel already running

Starting VE ...

================================ begin /etc/vz/conf/1.mount =============================

Removing and repopulating: root etc var tmp dev

Mounting: /var/lib/vz/template/debian-5.0-amd64-maxwell home apps
----------------------------- end /etc/vz/conf/1.mount -----------------------------

VE is mounted
Setting CPU units: 1000
Configure meminfo: 2000000
VE start in progress...
TIME: 0 seconds.
Waiting for container to finish booting.
/usr/lib/mw/startxvnc: Becoming nobody.
/usr/lib/mw/startxvnc: Waiting for 8-byte vncpasswd and EOF.
1+0 records in
1+0 records out
8 bytes (8 B) copied, 3.5333e-05 s, 226 kB/s
Got the vncpasswd
Adding auth for 10.51.0.1:0 and 10.51.0.1/unix:0
xauth: creating new authority file Xauthority-10.51.0.1:0
Adding IP address(es): 10.51.0.1
if-up.d/mountnfs[venet0]: waiting for interface venet0:0 before doing
NFS mounts (warning).
WARNING: Settings were not saved and will be resetted to original valu
es on next start (use --save flag)

# vzlist

<table>
<thead>
<tr>
<th>VEID</th>
<th>NPROC</th>
<th>STATUS</th>
<th>IP_ADDR</th>
<th>HOSTNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>running</td>
<td>10.51.0.1</td>
<td>-</td>
</tr>
</tbody>
</table>

# openssl s_client -connect localhost:4001

This should report an SSL connection with a self signed certificate and output text should end with:

---
RFB 003.008
If you see this then you successfully connected to the VNC server running inside the newly created OpenVZ session.

Clean up

```bash
# maxwell_service stopvnc 1
```

Which should give output similar to:

```
Killing 6 processes in veid 1 with signal 1
Killing 7 processes in veid 1 with signal 2
Killing 5 processes in veid 1 with signal 15
Got signal 9
Stopping VE ...
VE was stopped
-begin /etc/vz/conf/1.umount -

Unmounting /var/lib/vz/root/1/usr
Unmounting /var/lib/vz/root/1/home
Unmounting /var/lib/vz/root/1/apps
Unmounting /var/lib/vz/root/1/.root

Removing /var/lib/vz/root/1 :root etc var tmp dev/shm dev
Removing /var/lib/vz/private/1: apps bin emul home lib lib32 lib64 mnt
    opt proc sbin sys usr .root
-end /etc/vz/conf/1.umount -

VE is unmounted
```
Maxwell Client

Install

# apt-get install -y hubzero-mw-client

Configure

# hzcms configure mw-client --enable

Test

# su www-data
$ ssh -i /etc/mw-client/maxwell.key root@localhost ls
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is e5:3c:7d:41:71:0b:0f:2a:0c:0e:bb:15:4d:e7:2f:08.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
list of files
$ exit
#
vncproxy

Install

# apt-get install -y hubzero-vncproxy

Configure

# hzcms configure vncproxy --enable
telequotad

install

# apt-get install -y hubzero-telequotad

Configure

In order for filesystems quotas to work they must be enabled when they are mounted. Determine which filesystem contains your home directories and add "quota" to the mount option of the corresponding entry in the /etc/fstab file. Only the filesystem with /home on it matters to telequotad.

If quotas weren't already in affect, the run something like the following (depending on your filesystem configuration) to start up the quota system.

# mount -oremount /
# /etc/init.d/quota restart

Test

# repquota -a

Should show disk usage for all users.
Workspace

Install

# apt-get install hubzero-app
# apt-get install hubzero-app-workspace
# hubzero-app install --publish /usr/share/hubzero/apps/workspace-1.3.hza

Test

You should then be able to log in to the site and see the "Workspace" tool in the tool list and launch it in your browser.
Metrics

Install

# apt-get install hubzero-metrics

Configure

# hzcms configure metrics --enable
Rappture

Install

# apt-get install hubzero-rappture

Configure

Rappture is used from inside a container and needs several other packages installed to allow use of all its features. This process has been simplified by using the hubzero-rappture-session with only contains the dependencies needed to pull in these other packages.

# chroot /var/lib/vz/template/debian-6.0-amd64-maxwell
# apt-get update
# apt-get upgrade
# apt-get install hubzero-rappture-session
# exit

A workspace may need to be opened and closed a few times before the changes to the session template appear in a workspace.

Test

A user must setup their runtime environment in order to use the Rappture toolkit. Run the following command before attempting to run any Rappture tests.

use rappture

Rappture comes with several demonstration scripts that can effectively test many parts of the package. These demonstrations must be copied to a user's home directory within a workspace before running.

$ mkdir examples
$ cp -r /apps/share/rappture/examples/* examples/.
$ cd examples
$ ./demo.bash
A window should open on the workspace showing that part of the demonstration. Close that window to see the next demonstration. Some demonstrations may need something inputted to work properly (such as the graphing calculator).
Filexfer

Install

# apt-get install -y hubzero-filexfer-xlate

Configure

# hzcms configure filexfer --enable
Submit

Introduction

The submit command provides a means for HUB end users to execute applications on remote resources. The end user is not required to have knowledge of remote job submission mechanics. Jobs can be submitted to traditional queued batch systems including PBS and Condor or executed directly on remote resources.

Installation

```
# apt-get install hubzero-submit-pegasus
# apt-get install hubzero-submit-condor
# apt-get install hubzero-submit-common
# apt-get install hubzero-submit-server
# apt-get install hubzero-submit-distributor
# apt-get install hubzero-submit-monitors
# hzcms configure submit-server --enable
# /etc/init.d/submit-server start
```

At completion of the apt-get install commands several files will be located in the directory /opt/submit. Excluding python files the directory listing should like the following:

```
ls -a -I "[A-Z]*.py" -I "[A-Z]*.pyc" -I "." -I "." 
_batch
Batches monitors   distributor.sh monitorJob.py monitorTunnelA.py server.py
Files             environmentWhitelist.dat monitorJobDB monitorTunnelD.py sites.dat
                  environmentWhitelist.dft monitorJobQ.py monitorTunnelI.py sites.dft
                  etc monitorJobR.py monitorTunnelR.py tools.dat
                  managers.dat monitorJobS.py monitorTunnelT.py tools.dft
                  distributors.dat monitorJobT.py monitors.dat tunnels.dat
```

Configuration

submit provides a mechanism to execute jobs on machines outside the HUB domain. To accomplish this feat some configuration is required on the HUB and some additional software must be installed and configured on hosts in remote domains. Before attempting to configure submit it is necessary to obtain access to the target remote domain(s). The premise is that a single account on the remote domain will serve as an execution launch point for all HUB end...
users. It is further assumed that access to this account can be made by direct ssh login or using an ssh tunnel (port forwarding).

Having attained account access to one or more remote domains it is possible to proceed with submit configuration. To get started the ssh public generated by the installation should be transferred to the remote domain host(s).

**HUB Configuration**

The behavior of submit is controlled through a set of configuration files. The configuration files contain descriptions of the various parameters required to connect to a remote domain, exchange files, and execute simulation codes. There are separate files for defining remote sites, staged tools, multiprocessor managers, file access controls, permissible environment variables, remote job monitors, and ssh tunneling. Most parameters have default values and it is not required that all parameters be explicitly defined in the configuration files. A simple example is given for each category of configuration file.
Remote sites are defined in the file sites.dat. Each remote site is defined by a stanza indicating an access mechanism and other account and venue specific information. Defined keywords are

- [name] - site name. Used as command line argument (-v/--venue) and in tools.dat (destinations)
- venues - comma separated list of hostnames. If multiple hostnames are listed one site will chosen at random.
- tunnelDesignator - name of tunnel defined in tunnels.dat.
- siteMonitorDesignator - name of site monitor defined in monitors.dat.
- venueMechanism - possible mechanisms are ssh and local.
- remoteUser - login user at remote site.
- remoteBatchAccount - some batch systems requirement that an account be provided in addition to user information.
- remoteBatchSystem - the possible batch submission systems include CONDOR, PBS, SGE, and LSF. SCRIPT may also be specified to specify that a script will be executed directly on the remote host.
- remoteBatchQueue - when remoteBatchSystem is PBS the queue name may be specified.
- remoteBatchPartition - slurm parameter to define partition for remote job
- remoteBatchPartitionSize - slurm parameter to define partition size, currently for BG machines.
- remoteBatchConstraints - slurm parameter to define constraints for remote job
- parallelEnvironment - sge parameter
- remoteBinDirectory - define directory where shell scripts related to the site should be kept.
- remoteApplicationRootDirectory - define directory where application executables are located.
- remoteScratchDirectory - define the top level directory where jobs should be executed. Each job will create a subdirectory under remoteScratchDirectory to isolated jobs from each other.
- remotePpn - set the number of processors (cores) per node. The PPN is applied to PBS and LSF job description files. The user may override the value defined here from the command line.
- remoteManager - site specific multi-processor manager. Refers to definition in
• remoteHostAttribute - define host attributes. Attributes are applied to PBS description files.
• stageFiles - A True/False value indicating whether or not files should be staged to remote site. If the job submission host and remote host share a file system file staging may not be necessary. Default is True.
• passUseEnvironment - A True/False value indicating whether or not the HUB 'use' environment should passed to the remote site. Default is False. True only makes sense if the remote site is within the HUB domain.
• arbitraryExecutableAllowed - A True/False value indicating whether or not execution of arbitrary scripts or binaries are allowed on the remote site. Default is True. If set to False the executable must be staged or emanate from /apps. (deprecated)
• executableClassificationsAllowed - classifications accepted by site. Classifications are set in appaccess.dat
• members - a list of site names. Providing a member list gives a layer of abstraction between the user facing name and a remote destination. If multiple members are listed one will be randomly selected for each job.
• state - possible values are enabled or disabled. If not explicitly set the default value is enabled.
• failoverSite - specify a backup site if site is not available. Site availability is determined by site probes.
• checkProbeResult - A True/False value indicating whether or not probe results should determine site availability. Default is True.
• restrictedToUsers - comma separated list of user names. If the list is empty all users may garner site access. User restrictions are applied before group restrictions.
• restrictedToGroups - comma separated list of group names. If the list is empty all groups may garner site access.
• logUserRemotely - maintain log on remote site mapping HUB id, user to remote batch job id. If not explicitly set the default value is False.
• undeclaredSiteSelectionWeight - used when no site is specified to choose between sites where selection weight > 0.
• minimumWallTime - minimum walltime allowed for site or queue. Time should be expressed in minutes.
• maximumWallTime - maximum walltime allowed for site or queue. Time should be expressed in minutes.
• minimumCores - minimum number of cores allowed for site or queue.
• maximumCores - maximum number of cores allowed for site or queue.
• pegasusTemplates - pertinent pegasus templates for site, rc, and transaction files.

An example stanza is presented for a site that is accessed through ssh.
Tools

Staged tools are defined in the file tools.dat. Each staged tool is defined by a stanza indicating an where a tool is staged and any access restrictions. The existence of a staged tool at multiple sites can be expressed with multiple stanzas or multiple destinations within a single stanza. If the tool requires multiprocessors a manager can also be indicated. Defined keywords are

- [name] - tool name. Used as command line argument to execute staged tools. Repeats are permitted to indicate staging at multiple sites.
- destinations - comma separated list of destinations. Destination may exist in sites.dat or be a grid site defined by a ClassAd file.
- executablePath - path to executable at remote site. The path may be given as an absolute path on the remote site or a path relative to remoteApplicationRootDirectory defined in sites.dat.
- restrictedToUsers - comma separated list of user names. If the list is empty all users may garner tool access. User restrictions are applied before group restrictions.
- restrictedToGroups - comma separated list of group names. If the list is empty all groups may garner tool access.
- environment - comma separated list of environment variables in the form e=v.
- remoteManager - tool specific multi-processor manager. Refers to definition in managers.dat. Overrides value set by site definition.
- state - possible values are enabled or disabled. If not explicitly set the default value is
An example stanza is presented for a staged tool maintained in the yourhub account on a remote site.

```
[earth]
destinations = cluster
executablePath = ${HOME}/apps/planets/bin/earth.x
remoteManager = mpich-intel

[sun]
destinations = cluster
executablePath = ${HOME}/apps/stars/bin/sun.x
remoteManager = mpich-intel
```

Monitors

Remote job monitors are defined in the file monitors.dat. Each remote monitor is defined by a stanza indicating where the monitor is located and to be executed. Defined keywords are

- [name] - monitor name. Used in sites.dat (siteMonitorDesignator)
- venue - hostname upon which to launch monitor daemon. Typically this is a cluster headnode.
- venueMechanism - monitoring job launch process. The default is ssh.
- tunnelDesignator - name of tunnel defined in tunnels.dat.
INSTALLATION

- remoteUser - login user at remote site.
- remoteBinDirectory - define directory where shell scripts related to the site should be kept.
- remoteMonitorCommand - command to launch monitor daemon process.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example stanza is presented for a remote monitor tool used to report status of PBS jobs.

```
[clusterPBS]
venue = cluster.campus.edu
remoteUser = yourhub
remoteMonitorCommand = '${HOME}/SubmitMonitor/monitorPBS.py'
```

Multi-processor managers

Multiprocessor managers are defined in the file managers.dat. Each manager is defined by a stanza indicating the set of commands used to execute a multiprocessor simulation run. Defined keywords are

- [name] - manager name. Used in sites.dat and tools.dat.
- computationMode - indicate how to use multiple processors for a single job. Recognized values are mpi, parallel, and matlabmpi. Parallel application request multiprocess have there own mechanism for inter process communication. Matlabmpi is used to enable the an Matlab implementation of MPI.
- preManagerCommands - comma separated list of commands to be executed before the manager command. Typical use of pre manager commands would be to define the environment to include a particular version of MPI amd/or compiler, or setup MPD.
- managerCommand - manager command commonly mpirun. It is possible to include strings that will be substituted with values defined from the command line.
- postManagerCommands - comma separated list of commands to be executed when the manager command completes. A typical use would be to terminate an MPD setup.
• mpiRankVariable - define environment variable set by manager command to define process rank. Recognized values are: MPIRUN_RANK, GMPI_ID, RMS_RANK, MXMPI_ID, MSTI_RANK, PMI_RANK, and OMPI_MCA_ns_nds_vpid. If no variable is given an attempt is made to determine process rank from command line arguments.
• environment - comma separated list of environment variables in the form e=v.
• moduleInitialize - initialize module script for sh
• modulesUnload - modules to be unloaded clearing way for replacement modules
• modulesLoad - modules to load to define mpi and other libraries
• state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example stanza is presented for a typical MPI instance. The given command should be suitable for /bin/sh execution.

[mpich-intel]
preManagerCommands = . ${MODULESHOME}/init/sh, module load mpich-intel/11.1.038
managerCommand = mpirun -machinefile ${PBS_NODEFILE} -np NPROCESSORS

The token NPROCESSORS is replaced by an actual value at runtime.

File access controls

Application or file level access control is described by entries listed in the file appaccess.dat. The ability to transfer files from the HUB to remote sites is granted on a group basis as defined by white and black lists. Each list is given a designated priority and classification. In cases where a file appears on multiple lists the highest priority takes precedence. Simple wildcard operators are allowed the in the filename declaration allowing for easy listing of entire directories. Each site lists acceptable classification(s) in sites.dat. Defined keywords are

• [group] - group name.
• whitelist - comma separated list of paths. Wildcards allowed.
• blacklist - comma separated list of paths. Wildcards allowed.
• priority - higher priority wins
• classification - apps or user. user class are treated are arbitrary executables.
• state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example file giving permissions reminiscent of those defined in earlier submit releases is presented here

```
[public]
whitelist = /apps/.*
priority = 0
classification = apps

[submit]
whitelist = ${HOME}/.*
priority = 0
classification = home
```

The group public is intended to include all users. Your system may use a different group such as users for this purpose. The definitions shown here allow all users access to files in /apps where applications are published. Additionally members of the submit group are allowed to send files from their $HOME directory.

**Environment variables**

Legal environment variables are listed in the file environmentwhitelist.dat. The objective is to prevent end users from setting security sensitive environment variables while allowing application specific variables to be passed to the remote site. Environment variables required to define multiprocessor execution should also be included. The permissible environment variables should be entered as a simple list - one entry per line. An example file allowing use of a variables used by openmp and mpich is presented here.

```
# environment variables listed here can be specified from the command line with -e/--env option. Attempts to specify other environment varia bles will be ignored and the values will not be passed to the remote s ite.

OMP_NUM_THREADS
MPICH_HOME
```
Tunnels

In some circumstances access to clusters is restricted such that only a select list of machines is allowed to communicate with the cluster job submission node. The machines that are granted such access are sometimes referred to as gateways. In such circumstances ssh tunneling or port forwarding can be used to submit HUB jobs through the gateway machine. Tunnel definition is specified in the file tunnels.dat. Each tunnel is defined by a stanza indicating gateway host and port information. Defined keywords are

- [name] - tunnel name.
- venue - tunnel target host.
- venuePort - tunnel target port.
- gatewayHost - name of the intermediate host.
- gatewayUser - login user on gatewayHost.
- localPortOffset - local port offset used for forwarding. Actual port is localPortMinimum + localPortOffset

An example stanza is presented for a tunnel between the HUB and a remote venue by way of an accepted gateway host.

```
[cluster]
venue = cluster.campus.edu
venuePort = 22
gatewayHost = gateway.campus.edu
gatewayUser = yourhub
localPortOffset = 1
```
Initialization Scripts and Log Files

The submit server and job monitoring server must be started as daemon processes running on the submit host. If ssh tunneling is going to be used an addition server must be started as a daemon process. Each daemon process writes to a centralized log file facilitating error recording and debugging.

Initialize daemon scripts

Scripts for starting the server daemons are provided and installed in /etc/init.d. The default settings for when to start and terminate the scripts are adequate.

Log files

Submit processes log information to files located in the /var/log/submit directory tree. The exact location varies depending on the vintage of the installation. Each process has its own log file. The three most important log files are submit-server.log, distributor.log, and monitorJob.log.

submit.log

The submit-server.log file tracks when the submit server is started and stopped. Each connection from the submit client is logged with the command line and client ip address reported. All log entries are timestamped and reported by submit-server process ID (PID) or submit ID (ID:) once one has been assigned. Entries from all jobs are simultaneously reported and intermingled. The submit ID serves as a good search key when tracing problems. Examples of startup, job execution, and termination are given here. The job exit status and time metrics are also recorded in the MySQL database JobLog table.

```
[Sun Aug 26 17:28:24 2012] 0: ########################################
[Sun Aug 26 17:28:24 2012] 0: Listening: protocol='tcp', host='', port =830

[Sun Sep 23 12:33:28 2012] 0: Server will time out in 60 seconds.
[Sun Sep 23 12:33:28 2012] 0: Cumulative job load is 0.84.  (Max: 510.00)
[Sun Sep 23 12:33:28 2012] 1670: Args are:['/usr/bin/submit', '--local', '-p', '@@iv=-3:1.5:3', '/home/hubzero/user/hillclimb/bin/hillclimb1
```
.py', 'seed', '10', 'initialvalue', '@@iv', 'lowerbound', '-3', 'upperbound', '3', 'function', 'func2', 'solutionslog', 'solutions.dat', 'bestresultlog', 'best.dat']

[Sun Sep 23 12:33:38 2012] 0: Server will time out in 60 seconds.
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=1:local status=0 cpu=0.030000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=2:local status=0 cpu=0.040000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=3:local status=0 cpu=7.050000 real=7.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=4:local status=0 cpu=0.080000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=5:local status=0 cpu=0.020000 real=1.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue= status=0 cpu=10.428651 real=9.561828 wait=0.000000
[Sun Sep 23 12:48:44 2012] (1154) =================================================

[Sun Aug 26 17:28:17 2012] 0: Server(10836) was terminated by a signal 2.

distributor.log

The distributor.log file tracks each job as it progresses from start to finish. Details of remote site assignment, queue status, exit status, and command execution are all reported. All entries are timestamped and reported by submit ID. The submit ID serves as the key to join data reported in submit-server.log. An example for submit ID 1659 is listed here. Again the data for all jobs are intermingled.

[Sun Sep 23 00:04:21 2012] 0: quotaCommand = quota -w | tail -n 1
[Sun Sep 23 00:04:21 2012] 1659: command = tar vchf 0001659_01_input.
tar --exclude='*.svn*' -C /home/hubzero/user/data/sessions/3984L .__local_jobid.00001659_01 sayhiinquire.dax

[Sun Sep 23 00:04:21 2012] 1659: remoteCommand pegasus-plan --dax ./sayhiinquire.dax

[Sun Sep 23 00:04:21 2012] 1659: workingDirectory /home/hubzero/user/data/sessions/3984L

[Sun Sep 23 00:04:21 2012] 1659: command = tar vrhf 00001659_01_input.tar --exclude='*.svn*' -C /home/hubzero/user/data/sessions/3984L/00001659/01 00001659_01.sh

[Sun Sep 23 00:04:21 2012] 1659: command = nice -n 19 gzip 00001659_01_input.tar

[Sun Sep 23 00:04:21 2012] 1659: command = /opt/submit/bin/receiveInput.sh /home/hubzero/user/data/sessions/3984L/00001659/01 /home/hubzero/user/data/sessions/3984L/00001659/01/.__timestamp_transferred.00001659_01

[Sun Sep 23 00:04:21 2012] 1659: command = /opt/submit/bin/submitbatchjob.sh /home/hubzero/user/data/sessions/3984L/00001659/01 ./00001659_01.pegasus

[Sun Sep 23 00:04:23 2012] 1659: remoteJobId = 2012.09.23 00:04:22.996 EDT: Submitting job(s).

2012.09.23 00:04:23.002 EDT: 1 job(s) submitted to cluster 946.

2012.09.23 00:04:23.007 EDT: ---------------------------------------

2012.09.23 00:04:23.012 EDT: File for submitting this DAG to Condor:

: sayhi_inquire-0.dag.condor.sub

2012.09.23 00:04:23.023 EDT: Log of DAGMan debugging messages

: sayhi_inquire-0.dag.dagman.out

2012.09.23 00:04:23.028 EDT: Log of Condor library output

: sayhi_inquire-0.dag.lib.out

2012.09.23 00:04:23.033 EDT: Log of Condor library error messages

: sayhi_inquire-0.dag.lib.err

2012.09.23 00:04:23.038 EDT: Log of the life of condor_dagman itself

: sayhi_inquire-0.dag.dagman.log

2012.09.23 00:04:23.044 EDT:

2012.09.23 00:04:23.049 EDT: ---------------------------------------

2012.09.23 00:04:23.054 EDT: Your Workflow has been started and runs in base directory given below

2012.09.23 00:04:23.059 EDT: cd /home/hubzero/user/data/sessions/3984L/00001659/01/work/pegasus

2012.09.23 00:04:23.075 EDT: *** To monitor the workflow you can run ***
2012.09.23 00:04:23.085 EDT:  
2012.09.23 00:04:23.090 EDT:  pegasus-status -l /home/hubzero/user/data/sessions/3984L/00001659/01/work/pegasus  
2012.09.23 00:04:23.096 EDT:  
2012.09.23 00:04:23.101 EDT:  *** To remove your workflow run ***  
2012.09.23 00:04:23.106 EDT:  pegasus-remove /home/hubzero/user/data/sessions/3984L/00001659/01/work/pegasus  
2012.09.23 00:04:23.111 EDT:  
2012.09.23 00:04:23.117 EDT:  Time taken to execute is 0.993 seconds  
[Sun Sep 23 00:04:23 2012] 1659: status:Job N WF-DiaGrid  
[Sun Sep 23 00:04:38 2012] 1659: status:DAG R WF-DiaGrid  
[Sun Sep 23 00:10:42 2012] 0: quotaCommand = quota -w | tail -n 1  
[Sun Sep 23 00:10:42 2012] 1660: command = tar vchf 00001660_01_input.tar --exclude='*.svn*' -C /home/hubzero/clarksm .__local_jobid.00001660_01 noerror.sh  
[Sun Sep 23 00:10:42 2012] 1660: remoteCommand ./noerror.sh  
[Sun Sep 23 00:10:42 2012] 1660: workingDirectory /home/hubzero/clarksm  
[Sun Sep 23 00:10:42 2012] 1660: command = tar vrhf 00001660_01_input.tar --exclude='*.svn*' -C /home/hubzero/clarksm/00001660/01 00001660_01.sh  
[Sun Sep 23 00:10:42 2012] 1660: command = nice -n 19 gzip 00001660_01_input.tar  
[Sun Sep 23 00:10:42 2012] 1660: command = /opt/submit/bin/receiveinput.sh /home/hubzero/clarksm/00001660/01 /home/hubzero/clarksm/00001660/01/__timestamp_transferred.00001660_01  
[Sun Sep 23 00:10:42 2012] 1660: command = /opt/submit/bin/submitbatchjob.sh /home/hubzero/clarksm/00001660/01 ./00001660_01.condor  
[Sun Sep 23 00:10:42 2012] 1660: remoteJobId = Submitting job(s). 1 job(s) submitted to cluster 953.  
[Sun Sep 23 00:10:42 2012] 1660: status:Job N DiaGrid  
[Sun Sep 23 00:11:47 2012] 1660: status:Simulation I DiaGrid  
[Sun Sep 23 00:12:07 2012] 1660: Received SIGINT!  
[Sun Sep 23 00:12:07 2012] 1660: waitForBatchJobs: nCompleteRemoteJobIndexes = 0, nIncompleteJobs = 1, abortGlobal = True  
[Sun Sep 23 00:12:07 2012] 1660: command = /opt/submit/bin/killbatchjob.sh 953.0 CONDOR  
[Sun Sep 23 00:12:07 2012] 1660: Job 953.0 marked for removal  
[Sun Sep 23 00:12:07 2012] 1660: status:Simulation I DiaGrid  
[Sun Sep 23 00:12:52 2012] 1660: status:Simulation D DiaGrid  
[Sun Sep 23 00:12:52 2012] 1660: venue=1:localCONDOR:953.0:DiaGrid status=258 cputime=0.000000 realtime=0.000000 waittime=0.000000 ncpus=1  
[Sun Sep 23 00:28:14 2012] 1659: status:DAG D WF-DiaGrid
monitorJob.log

The monitorJob.log file tracks the invocation and termination of each remotely executed job monitor. The remote job monitors are started on demand when jobs are submitted to remote sites. The remote job monitors terminate when all jobs complete at a remote site and no new activity has been initiated for a specified amount of time - typically thirty minutes. A typical report should look like:

[Sun Aug 26 17:29:16 2012] (1485) ************************************
[Sun Aug 26 17:29:16 2012] (1485) * distributor job monitor started *
[Sun Aug 26 17:29:16 2012] (1485) ************************************
[Sun Aug 26 17:29:16 2012] (1485) loading active jobs
[Sun Aug 26 17:29:16 2012] (1485) 15 jobs loaded from DB file
[Sun Aug 26 17:29:16 2012] (1485) 15 jobs loaded from dump file
[Sun Aug 26 17:29:16 2012] (1485) 4 jobs purged
[Sun Aug 26 17:29:16 2012] (1485) 11 monitored jobs
[Sun Aug 26 18:02:04 2012] (24250) Launching wf-diagrid
[Sun Aug 26 18:02:04 2012] (1485) 12 monitored jobs
[Sun Aug 26 18:02:15 2012] (1485) Update message received from wf-diagrid
It is imperative that the job monitor be running in order for notification of job progress to occur. If
users report that their job appears to hang check to make sure the job monitor is running. If
necessary take corrective action and restart the daemon.

monitorTunnel.log

The monitorTunnel.log file tracks invocation and termination of each ssh tunnel connection. If
users report problems with job submission to sites accessed via an ssh tunnel this log file
should be checked for indication of any possible problems.

Remote Domain Configuration

For job submission to remote sites via ssh it is necessary to configure a remote job monitor and
a set of scripts to perform file transfer and batch job related functions. A set of scripts can be
used for each different batch submission system or in some cases they may be combined with
appropriate switching based on command line arguments. A separate job monitor is need for
each batch submission system. Communication between the HUB and remote resource via ssh
requires inclusion of a public key in the authorized_keys file.

Job monitor daemon
A remote job monitor runs a daemon process and reports batch job status to a central job monitor located on the HUB. The daemon process is started by the central job monitor on demand. The daemon terminates after a configurable amount of inactivity time. The daemon code needs to be installed in the location declared in the monitors.dat file. The daemon requires some initial configuration to declare where it will store log and history files. The daemon does not require any special privileges any runs as a standard user. Typical configuration for the daemon looks like this:

```
$ cat monitors.dat
[clusterPBS]
venue = cluster.campus.edu
remoteUser = yourhub
remoteMonitorCommand = $HOME/Submit/monitorPBS.py
```

The directory defined by MONITORLOGLOCATION needs to be created before the daemon is started. Sample daemon scripts used for PBS, LSF, SGE, Condor, Load Leveler, and Slurm batch systems are included in directory BatchMonitors.

**File transfer and batch job scripts**

The simple scripts are used to manage file transfer and batch job launching and termination. The location of the scripts is entered in sites.dat.
Examples scripts suitable for use with PBS, LSF, Condor, Load Leveler, and Slurm are included in directory Scripts. After modifications are made to monitors.dat the central job monitor must be notified. This can be accomplished by stopping and starting the submon daemon or a HUP signal can be sent to the monitorJob.py process.

File transfer - input files
Receive compressed tar file containing input files required for the job on stdin. The file transferredTimestampFile is used to determine what newly created or modified files should be returned to the HUB.

receiveinput.sh jobWorkingDirectory jobScratchDirectory transferredTimestampFile

Batch job script - submission
Submit batch job using supplied description file. If arguments beyond job working directory and batch description file are supplied an entry is added to the remote site log file. The log file provides a record relating the HUB end user to the remote batch job identifier. The log file should be placed at a location agreed upon by the remote site and HUB.

submitbatchjob.sh jobWorkingDirectory jobScratchDirectory jobDescriptionFile

The jobId is returned on stdout if job submission is successful. For an unsuccessful job submission the returned jobId should be -1.

File transfer - output files
INSTRUCTION

Return compressed tar file containing job output files on stdout.

transmitresults.sh jobWorkingDirectory

File transfer - cleanup
Remove job specific directory and any other dangling files

cleanupjob.sh jobWorkingDirectory jobScratchDirectory jobClass

Batch job script - termination
Terminate given remote batch job. Command line arguments specify job identifier and batch system type.

killbatchjob.sh jobId jobClass

Batch job script - post process
For some jobClasses it is appropriate to preform standard post processing actions. An example of such a jobClass is Pegasus.

postprocessjob.sh jobWorkingDirectory jobScratchDirectory jobClass

Access Control Mechanisms

By default tools and sites are configured so that access is granted to all HUB members. In some cases it is desired to restrict access to either a tool or site to a subset of the HUB membership. The keywords restrictedToUsers and restrictedToGroups provide a mechanism to apply restrictions accordingly. Each keyword should be followed by a list of comma separated values of userids (logins) or groupids (as declared when creating a new HUB group). If user or group restrictions have been declared upon invocation of submit a comparison is made between the restrictions and userid and group memberships. If both user and group restrictions are declared the user restriction will be applied first, followed by the group restriction.

In addition to applying user and group restrictions another mechanism is provided by the
executableClassificationsAllowed keyword in the sites configuration file. In cases where the executable program is not pre-staged at the remote sites the executable needs to be transferred along with the user supplied inputs to the remote site. Published tools will have their executable program located in the /apps/tools/revision/bin directory. For this reason submitted programs that reside in /apps are assumed to be validated and approved for execution. The same cannot be said for programs in other directories. The common case where such a situation arises is when a tool developer is building and testing within the HUB workspace environment. To grant a tool developer the permission to submit such arbitrary applications the site configuration must allow arbitrary executables and the tool developer must be granted permission to send files from their $HOME directory. Discrete permission can be granted on a file by file basis in appaccess.dat.