Setting up a system for data archiving using FTS3

Lydia Heck (ICC, Durham University)
Jens Jensen and Brian Davies (STFC)

February 1, 2016

Todo list

1 Introduction

In April 2015 DiRAC gave the mandate to the DiRAC site in Durham to setup the pilot for effective archiving/backup of the data on the Data Centric System to the File Service/Archiving system at RAL. Here is a description of the setup procedure and tools and software involved. In the venture Durham was helped by Jens Jensen and Brian Davies from RAL and information material interchanged by email has been worked into the document.

2 Certificates

All file transfer communications to RAL are handled via gridftp and access and authentication for Grid traffic is handled by personal and host grid certficates.

For a full configuration of a gridftp server a grid certificate is required from a grid authority, and in our case that would be UKCA (link below). To acquire a grid host certificate the person who applies for that certificate firstly requires a personal grid certificate. The personal grid certificate is also required to have access to the archiving storage and will be 'attached' to the GridPP DiRAC account.

To apply for a grid certificate go to the web page (figure 1)
`http://www.ngs.ac.uk/ukca/certificates/certwizard`.

Click on 'CLICK HERE to Launch CertWizard' and follow the menu. Should the tool not launch from the web pages, then the tool can be downloaded to start manually. Java 1.7.x as a minimum is required. Click the tab "Apply For/Manage Your Certificate" – the other tabs are not currently needed (not even the "Setup" one!)

The tool *CertWizard* is necessary for applying for and managing a personal certificate as well as applying for and managing a host certificate and/or for renewing certificates that are about to expire. (*CertWizard* can do much more , but that is not part of the present discussion.) Both the personal and host certificate are valid for 12 months from signing[1]. Support is available from support@grid-support.ac.uk.

When you apply for the *personal* certificate (also known as "user certificate") you will have to get your certificate request 'approved'. In the application process a list of approving authorities are being offered

[1]It is actually valid for 13 months, the idea being that it is renewed at the same time every year but to leave 30 days' notice on either side of its renewal date.

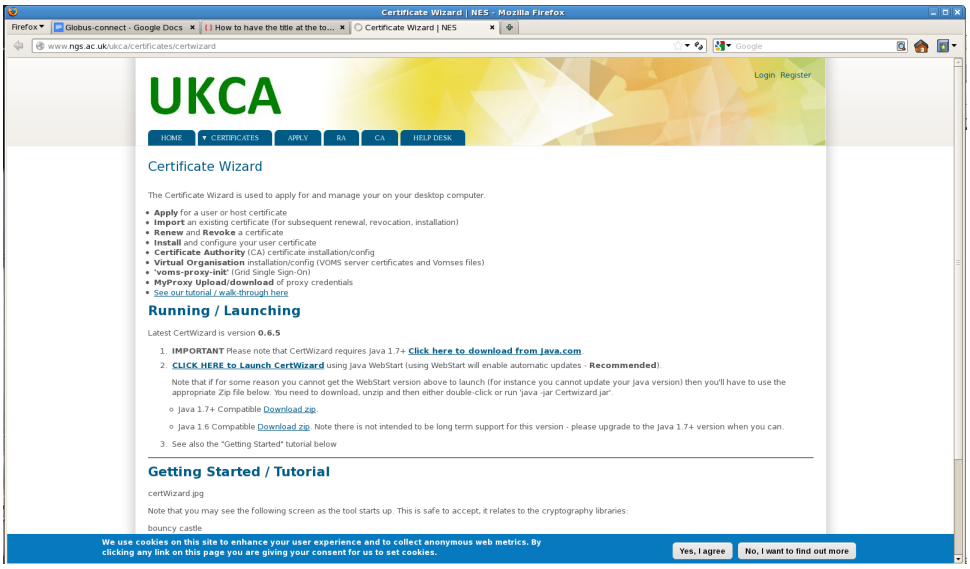

Figure 1: http://www.ngs.ac.uk/ukca/certificates/certwizard

and you might want to choose the closest, most convenient authority for you. This authority will be notified of your request. For them to authorize your certificate you will have to appear in person with an approved photo id to prove your identity[2]. They are obliged to take a copy of your photo id so that they can prove in an audit that they have followed the proper security procedures.

For a list of your nearest signing authority please check `https://portal.ca.grid-support.ac.uk/caportal/pub/viewralist`. For Cambridge there is one named person at UCS, for Durham there are 3 named people at the IPPP and for Edinburgh there are 4 at NeSC but for Leicester there is none.

Host certificates also need approval by the authority, and require that they check that the person requesting the host certificate is responsible for the host in question – however, this check is much more lightweight and might not need visiting the authority in person.

Once the certificate request has been approved, it needs to be signed by the actual Certification Authority. This typically happens within one working day (usually on the same day) as the approval, and the requestor is notified by the signing/issuance by email. *CertWizard* will in fact also notice automatically and refresh its database. *CertWizard* will be required to manage - export and save - the certificates.

It is recommended to make a backup of the encrypted key (by exporting the (encrypted) key and backing it up somewhere). Alternatively, a backup of the system where *CertWizard* is running will be sufficient.

2.1 Anatomy of a Certificate

If you are curious what's going on, read on – if you just want to put it to use, feel free to skip to the next section.

Slightly simplified, a certificate consists of a secret part ("private key") and a public part (the "certificate"); the public part is the one being requested, approved, and finally signed. Meanwhile the secret part stays on the system that generated the public part: since it is secret it is usually protected by a

[2]Passport and drivers licence (with photo) are always accepted (any country, not just UK) but university (or other workplace) photo id are preferred provided the authority can assess the validity (*i.e.* they know what a university photo id should look like) as these tend to contain much less personal data.

passphrase (passphrase protection is mandatory for keys corresponding to user certificates; requires at least 10 characters – host certificates are allowed to be protected by filesystems permissions only.)

When a client connects to a server, the server will send its certificate to the client, and the client checks that the server hostname matches the name used for the connection, it checks the *validity* of the certificate against a trusted CA, and then sends the server a challenge using the public key. The server uses its private key to respond to the challenge, sending the response to the client. Morever, we use *mutual authentication*, where both the client and the server send certificates to each other, and thus authenticate each other.

2.2 Deploying and Using Certificates

The user certificates as used by *CertWizard* are by default expected in the directory `${HOME}/.ca` of the user with the default filename `usercert.pem` together with the user key `userkey.pem`. The userkey file must be visible only to the user otherwise connection is refused. (*i.e* mode `rw-------` or even `r--------`.) For globus transactions the user key pair are expected to be in `${HOME}/.globus` (see the document on how I set up Globus Connect on COSMA).

For managing certificates the user certificate will have to be imported into the user's web browser profile. This can be on the user's desktop or workstation, and does not have to be run on the DiRAC site's data archiving system from which you intend to push data to RAL.

Once the user certificate has been obtained, the certificate has to be enrolled into the virtual organization (vo) – `vo.dirac.ac.uk` – using the web page
`https://voms.gridpp.ac.uk:8443/voms/vo.dirac.ac.uk/register`.
The user certificate will have to be loaded into the web browser before this page can be accessed. For *firefox* this is done by choosing `Edit --> Preferences -- Advanced --> Certificates --> Import`

The user certificate will have to have a Tier 1 user id assigned to it. There is a set of these, called dirac01, dirac02, etc., with dirac01 being assigned to Durham. This then gives access to the RAL CASTOR storage system to copy the files. We might want to have as many GridPP user ids as there are systems in DiRAC, with the names dirac01 (Durham), dirac02, dirac03, dirac04 and dirac05. The *same account* must be set up at the DiRAC site – it doesn't have to have the same uid or gid, just the same (Unix) name, because the VO mapping will map the name used in the certificate into the local account name[3].

The host certificate obtained for the DiRAC gridftp server must be installed in /etc/grid-security/ as `hostcert.pem` with permssions 644 and the `hostkey.pem` in the same location with permission 600. The key must be unencrypted, protected only by the filesystem permissions.

2.3 Proxy Certificates

A "proxy" is a placeholder credential which allows a service to act on behalf of another service[4]. A certificate credential (sec 2.1) consists of a public part (the certificate) and a private part (the private key). The thinking behind proxies is that:

- The user's private key (if the user is a person) must be protected with a passphrase, but one cannot bother the user every time the key is used since they would have to be present and type the passphrase every time they run a command line tool (for example). Thus, it makes sense to have an "activated" private key which is valid for a shortish while.

[3] Larger VOs use *pool accounts* so do not need to synchronise naming between sites, but with a limited number of DiRAC sites, the simpler naming scheme was preferred.

[4] Not to be confused with a web proxy or HTTP proxy.

- Often, a remote service needs to act on behalf of the user, *e.g.* to transfer files or store data. However, the user's own private key should not be used for this, as it increases the risk of a compromise of the key – and also the user is not as a rule able to activate the key on the remote service.
- For security reasons, private keys should never be transferred: if a service needs a private key, it should generate it itself and ask a suitable authority to sign certificates that assert the identity of the entity.
- Proxy certificates are certificates signed by the user's own certificate[5]. Normally users are not allowed to sign certificates themselves (if they could, they could sign anything they like), but the servers we are calling have an exemption that says if the proxy's name (described below) is "similar" to the user's name, the signature will be accepted.
- Proxy certificates, like user certificates, can contain extensions. Since proxy certificates are usually short lived (on the order of day to a week), the extensions are often used for authorisation attributes (authorisations are traditionally shorter lived than identities).
- Proxy certificates' lifetimes are typically limited by policy or administratively (configuration settings), or both. See section 2.3.3 on page 6.

Proxies are standardised in RFC3820. The RFC describes how a proxy's name should be derived from a user's name; this is done by adding `CN` (commonName) entries to the user's name, either `CN=proxy` for "old style" proxies or `CN=`(number) for newer (aka RFC style) proxies.

2.3.1 Proxy debugging

(feel free to skip this section, but it might be helpful for troubleshooting)

Proxy file location Where does the proxy live? For a local proxy (which is used *in lieu* of the user's private key which is passphrase protected), the filename is `/tmp/x509up_u`X where X is the user's Unix uid (in decimal, without leading zeros).

A *delegated* proxy file, *i.e.* one which has been created by a remote service, can live in any location: it depends on the service. The FTS server puts them in `/tmp` as well, but with more elaborate file names which are based on the DN, with the "difficult" characters escaped, and with some random stuff added. See also item 4.3 on the creation of proxies with/for FTS.

Anatomy of the proxy file

If you look inside the proxy, you will find certificates and keys encoded in PEM format: PEM was originally Privacy Enhanced Mail, a scheme which enabled mail (which notoriously silently corrupted characters outside a narrow US ASCII subset) to safely transfer certificates and signatures. PEM has the advantage that the start and end of each entry is clearly delineated so the entries can be concatenated.

The format of the proxy file is always as follows:

- The first entry is the proxy certificate itself: the "active" certificate which proxies the user.
- The following entry is always the private key (unencrypted, so protected only with the filesystem permissions) associated with the proxy certificate.

[5]Technically, signatures are actually made with the private key and validated with the certificate, but for convenience of expression we shall say "signed with a certificate," meaning "the signature was made with the private key corresponding to the certificate in question and should be validated using the certificate."

- The next entries are the proxy certificates that have signed the proxy certificate (if any). A proxy may delegate another proxy, in which case it is a proxy certificate which signs the delegated proxy certificate (see next section).
- At the end of the certificate chain, the user certificate appears.
- There is no entry for the authority that signed the user certificate: for security reasons, these must be installed on the server end (or wherever the proxy certificate connects to, where it is validated).

On the command line, one can list the contents of the local proxy file with the command `grid-proxy-info`.

2.3.2 Creation of a proxy, VOMS and MyProxy proxies

This section describes briefly the interactive creation of a proxy for the purposes of troubleshooting proxy files; for further details of proxy generation with respect to the use with FTS, please refer to item 4.3.

Assuming the user has set up their "home" credentials (in the `.globus` directory, sec. 2.2), the command `grid-proxy-init` will prompt the user for the passphrase and generate a simple local proxy. The proxy file will contain (1) the proxy certificate, (2) the proxy private key, and (3) the user certificate.

As mentioned above, proxies can contain authorisation attributes. In the Grid world, authorisation attributes always come from VOMS, the Virtual Organisation Membership Service (section 2.2). VOMS proxies (*i.e.* proxies with VOMS extensions) are created in a similar way from the user's "home" credentials:

```
voms-proxy-init -voms vo.dirac.ac.uk
```

Note the parameter to inform the command of the name of the DiRAC VO. The command will prompt for the passphrase and the output of this command should be:

```
Enter GRID pass phrase for this identity:
Contacting voms.gridpp.ac.uk:15511 [/C=UK/O=eScience/OU=Manchester/L=HEP/CN=voms.gridpp.ac.uk]
"vo.dirac.ac.uk"...
Remote VOMS server contacted succesfully.


Created proxy in /tmp/x509up_u33002.

Your proxy is valid until Fri Oct 23 22:50:22 BST 2015
```

If there is an error, it is likely to be due to (a) the user certificate and private key have not been deployed to the `.globus` location correctly (section 2.2), (b) incorrect passphrase for the user private key, or (c) VOMS has not been configured correctly on the system on which the command is run (section **??**).

Again the status of the proxy can be viewed with `grid-proxy-info`, or `voms-proxy-info -all` lists also the VOMS extensions. The proxy file contains the same local proxy as before, only with a much bigger proxy certificate as it will have the VOMS extensions inside it.

MyProxy At this point we are not planning to use MyProxy (because the current version of FTS does not seem to be able to use MyProxy). However, as it may be relevant to future work, or to the use of Globus Online, we give here a very quick introduction to MyProxy. As this is (currently) a working document, we will extend this section if/when we can start using MyProxy.

MyProxy is a credential repository written by NCSA. Its aim is to allow users to upload or create proxies in the repository, whence they can delegate further proxies, either directly by running a command line tool in the location where they want the proxy, or by granting rights to a remote service (such as FTS, if FTS had still supported it) to automatically get or renew proxies from the MyProxy server (without the user's intervention).

2.3.3 Certificate and Proxy Lifetimes

Certificates typically (for the UK e-Science CA) live 13 months with the intention that the effective lifespan of the certificate should be precisely a year (this allows the diligent sysadmin or user to remember the "birthday" or "anniversary" of their host or personal certificates and renew them on time). A distinguishing feature of these certificates is that they can be *revoked* by the CA. Certificates that cannot be revoked – such as proxy certificates – must, by policy, have a short lifetime, the upper limit for which semi-arbitrarily has been set to 10^6 seconds (about 10-11 days, again allowing a weekly renewal plus a bit of extra time to cover bank holidays or absences etc.)

The `grid-proxy-init` command will in fact quite happily generate a proxy with an arbitrarily long lifetime. The implication of this is (a) it is a useful workaround allowing the certificate to be used (by using the proxy) for the whole lifetime of the certificate, but (b) if the proxy's (unprotected except by filesystem permissions) private key is compromised, the user certificate will have to be revoked; and (c) as this is a "plain" proxy it contains no VOMS extensions.

A proxy certificate is only valid as long as *every* certificate in its chain (issuing authority → user certificate → proxy → ... → proxy) is valid. Moreover, if any one of the proxies contains VOMS extensions, these extensions, too, can expire. To confuse matters further, many credential handling services (such as VOMS, MyProxy, or FTS) are configured to limit the lifetimes of the proxies that they create, silently truncating the requested lifetime if they think it is too long. Safe (but not always sufficiently useful) values for local proxies are 24 hours, for MyProxy proxies one week.

3 Setting up and configuring a system to archive to RAL

Before I configured the system for the archiving precedure to RAL, I had installed and configured Globus-Connect and the gridftp tools were installed as part of that procedure. I have documented that procedure in a separate document.

There are other packages that are required for running the system as the file transfer client to RAL and in the following I am trying to describe them as well.

3.1 Globus - Gridftp

Globus Toolkit, downloadable from `http://http://toolkit.globus.org/toolkit/` provides the following packages through the `globus-connect-server-repo-latest.noarch` repository – see `http://toolkit.globus.org/toolkit/docs/latest-stable/admin/install/`, section 1.1 *Prerequisites* (version numbers of the packages may change).

```
globus-gridmap-verify-myproxy-callout-2.7-2.el6.x86_64
gridftp-ifce-2.3.1-1.el6.x86_64
globus-gridftp-server-control-3.6-2.el6.x86_64
gridsite-libs-2.2.5-2.el6.x86_64
gridsite-2.2.5-2.el6.x86_64
globus-gridmap-eppn-callout-1.7-3.el6.x86_64
```

```
globus-gridftp-server-progs-7.26-1.el6+gt6.x86_64
globus-gridftp-server-7.26-1.el6+gt6.x86_64
globus-gridmap-callout-error-2.4-2.el6.x86_64
gfal2-plugin-gridftp-2.9.1-1.el6.x86_64
```

3.2 Certification Agencies = Trust Anchors

As all of grid access requires certification, *Certification Authorities* must be known on the system.

The trustanchors can be installed from the EGI-trustanchors repository (`https://wiki.egi.eu/wiki/EGIi_IGTF_Release`)

```
[root@data yum.repos.d]# cat egi-trustanchors.repo
[EGI-trustanchors]
name=EGI-trustanchors
baseurl=http://repository.egi.eu/sw/production/cas/1/current/
enabled=1
gpgcheck=1
gpgkey=http://repository.egi.eu/sw/production/cas/1/GPG-KEY-EUGridPMA-RPM-3
```

and the packages that are required are:

```
ca-policy-egi-core.noarch
ca-policy-lcg.noarch
lcg-CA.noarch
```

lcg-CA.noarch will install 100 ca_ packages for the individual certification authorities and hundreds of files are installed into the directory `/etc/grid-security/certificates`.

3.3 LCG software

Next install the LCG software packages and dependencies from the EPEL repository. For setting up the EPEL repository for CentOS 6.x in your yum configuration do

```
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -Uvh epel-release-6-8.noarch.rpm
```

The packages are:
lcg-util-libs
lcgdm-libs
fetch-crl

The **fetch-crl** tool has to be enabled using the command:

```
chkconfig --level 345 fetch-crl on
service fetch-crl start
```

This will start a cronjob. Please note, you might get warning messages from the system via email (if set up) from time to time from fetch-crl, about missing certificates. These can be (usually) ignored.

The fetch-crl tool will update the CRLs (Certificate Revocation Lists, in `/etc/grid-security/certificates`), which are needed because otherwise the Globus code will (likely) refuse to accept the certificates (CRLs typically have a `.r0` extension in this context).

3.4 FTS3

FTS3 packages provide the commands that will enable the transfer of data from the DiRAC site to the RAL archiving system. The installation and configuration of FTS3 are fully documented in the Administration Guide

`https://svnweb.cern.ch/trac/fts3/wiki/AdminGuide`

The FTS3 repository can be added to /etc/yum.repos.d/ creating a file, *fts3.repo* with the following content:

```
[FTS3]
name=FTS3
baseurl=http://grid-deployment.web.cern.ch/grid-deployment/dms/fts3/repos/el6/x86_64
enabled=1
gpgcheck=0
priority=15
```

As there is no server being set up, the only packages that are required `fts-client` and dependencies using the command `yum install fts-client` which will result in the installation of

```
fts-server-3.3.x-y.x86_64
fts-client-3.3.x-y.x86_64
fts-libs-3.3.x-y.x86_64
fts-mysql-3.3.x-y.x86_64
fts-python-3.3.x-y.x86_64
fts-monitoring-3.3.x-y.el6.noarch
```

(Of course the version numbers might change).

It is important to notice that in Durham on two separate occasions that after an upgrade fts-xyz-3.4.x was installed. With these version the connection to the FTS servers had RAL was not possible.

3.5 EMI software repositories

EMI: European Middleware Initiative and the repositories are hosted by CERN. They are:

`emi3-base` and `emi3-updates`, from which tools like edg-mkgridmap are installed and that have the repo files: `emi3.repo`

```
[emi3-base]
gpgkey=http://emisoft.web.cern.ch/emisoft/dist/EMI/3/RPM-GPG-KEY-emi
enabled=1
baseurl=http://emisoft.web.cern.ch/emisoft/dist/EMI/3/sl6/$basearch/base
gpgcheck=1
protect=1
priority=40
name=EMI 3 base


[emi3-updates]
gpgkey=http://emisoft.web.cern.ch/emisoft/dist/EMI/3/RPM-GPG-KEY-emi
enabled=1
baseurl=http://emisoft.web.cern.ch/emisoft/dist/EMI/3/sl6/$basearch/updates
gpgcheck=1
protect=1
priority=40
name=EMI 3 updates
```

and `emi3-third-party.repo`

```
[emi3-third-party]
gpgkey=http://emisoft.web.cern.ch/emisoft/dist/EMI/3/RPM-GPG-KEY-emi
enabled=1
```

```
baseurl=http://emisoft.web.cern.ch/emisoft/dist/EMI/3/sl6/$basearch/third-party
gpgcheck=1
protect=1
priority=40
name=EMI 3 third-party
```

3.6 edg-mkgridmap

From the emi3 repositories install `edg-mkgridmap` providing the command `edg-mkgridmap`. The command requires an input file `/etc/edg-mkgridmap.conf`
which for COSMA contains the entry:
`group vomss://voms.gridpp.ac.uk:8443/voms/vo.dirac.ac.uk/ dirac01`

(Note the "vomss," meaning "secure voms" analogously to https or ldaps.) For different DiRAC sites the configuration file refer to the site specific DiRAC account, which will be one of `dirac01 - dirac05`

Running the command in verbose mode shows the following output:

```
 edg-mkgridmap --verbose --output=/etc/grid-security/grid-mapfile


Loading configuration from /etc/edg-mkgridmap.conf

Operating configuration
GROUP          : vomss://voms.gridpp.ac.uk:8443/voms/vo.dirac.ac.uk/ dirac01
ACL            : allow *
DEFAULT_LCLUSER: .

Loading certificate subjects from vomss://voms.gridpp.ac.uk:8443/voms/vo.dirac.ac.uk/
"/C=UK/O=eScience/OU=Durham/L=eScience/CN=lydia heck" allowed by rule 'allow *'
"/C=UK/O=eScience/OU=CLRC/L=RAL/CN=brian davies" allowed by rule 'allow *'
"/C=UK/O=eScience/OU=CLRC/L=RAL/CN=jens sha2 jensen" allowed by rule 'allow *'
"/C=UK/O=eScience/OU=Durham/L=eScience/CN=lydia heck/CN=Robot:GridClient"
allowed by rule 'allow *'

Writing output to /etc/grid-security/grid-mapfile.0
Deleting /etc/grid-security/grid-mapfile.1
Linking /etc/grid-security/grid-mapfile to /etc/grid-security/grid-mapfile.1
Moving /etc/grid-security/grid-mapfile.0 to /etc/grid-security/grid-mapfile
```

with the content of `/etc/grid-security/grid-mapfile`

```
"/C=UK/O=eScience/OU=CLRC/L=RAL/CN=brian davies" dirac01
"/C=UK/O=eScience/OU=CLRC/L=RAL/CN=jens sha2 jensen" dirac01
"/C=UK/O=eScience/OU=Durham/L=eScience/CN=lydia heck" dirac01
"/C=UK/O=eScience/OU=Durham/L=eScience/CN=lydia heck/CN=Robot:GridClient"
dirac01
```

with a listing of several user mappings to the account dirac01: Brian Davies, Jens Jensen and Lydia Heck - the latter with two certificates. `/C=UK/O=eScience/OU=CLRC/L=RAL/CN=`
and `/C=UK/O=eScience/OU=Durham/L=eScience/CN` identify the certificate signing authorities for the different certficiates and are from the certificate headers: CLRC RAL and Durham eScience. Other accounts can/will be mapped when a user enrols and is accepted into the DiRAC vo.

3.7 Also required and installed from the emi3 repositories are:

```
voms-api-java3-3.0.5-1.sl6.noarch
voms-2.0.12-3.el6.x86_64
voms-clients3-3.0.6-1.el6.noarch
```

providing commands like `voms-proxy-init, voms-proxy-info, voms-proxy-destroy` and more. These are required for creating and maintaining grid proxies enabling the authorization proccess for archiving job submissions.

3.8 Firewall Rules

For using the fts-transfer-xxxx commands, the required ports on the data system are tcp 2811; and 50000-51000 tcp AND udp with the rules being (Please note: the interface name 'eth2' will have to be modified according the interface on the servers of the different DiRAC sites).

```
# GridPP rules
-A INPUT -i eth2 -p tcp -m state --state NEW -m tcp --dport 50000:51000 -j ACCEPT
-A INPUT -i eth2 -p udp -m state --state NEW -m udp --dport 50000:51000 -j ACCEPT
-A INPUT -i eth2 -s 130.246.176/22 -p tcp -m state --state NEW -m tcp --dport 2811 -j ACCEPT
-A INPUT -i eth2 -s 130.246.180/22 -p tcp -m state --state NEW -m tcp --dport 2811 -j ACCEPT
-A INPUT -i eth2 -s 130.246.221.148/32 -p tcp -m state --state NEW -m tcp --dport 2811 -j ACCEPT
```

4 Archiving data to RAL

A *job* is a request to the FTS server to transfer one or more files from a given location (here, a DiRAC site) to another (RAL). Each job is submitted to FTS using a proxy (see item 4.3 for the gory details), so a valid proxy must be available before submitting the job. A job is submitted on the command line using tools described below; the tools can then be used to track the status of the job itself as well as the transfers of the individual files in the job.

In general FTS will then schedule, execute, and monitor (and retry) the transfers, using the proxy that was delegated to the server.

Before we go into the details, it is worth remarking a few oddities of FTS:

- When the job is submitted, a proxy may be delegated with it (see item 4.3); if this proxy times out, the transfers will fail. A later proxy replacing the first one will not be used: once the job has been submitted, the proxy that the job will use is fixed.
- Zero length files *will not* be transferred and will appear in the list as failed transfers. This is (possibly) because a zero length file can indicate a read error on the source filesystem and needs investigating. It will be necessary to make a list of the zero length files and create them "by hand" (*i.e.* log into a RAL UI and create them with `nstouch`).
- Files will by default not be overwritten[6]. If the destination file exists, FTS will by default refuse the transfer and list it as failed. The `-o` switch will tell FTS to overwrite destination files (the `-o` switch is common to all files in a job).
- The default retry number is 1. In other words, FTS will never retry, by default; it will attempt the file transfer once and if it fails, mark it as failed (and if it should not have failed, it is then up to the submitter to resubmit this particular transfer in another job). The rationale for this was that in practice, when the default was 3, FTS spent a lot of time retrying hopeless cases, so it was considered better to report the failures and let someone investigate.

[6]The rationale for this behaviour is, for once, clear; most science data generated is write-once-read-many and data should not be overwritten; any derived data which is worth keeping will itself be placed in individual files and directories and then transferred, and temporary output files will in general never be transferred.

- We speculate (but haven't fully tested) that zero length and destination-exists transfers will also be retried if retry is used.
- The `--nostreams` switch does not mean "no streams," but rather "number of streams" (to use in parallel transfers).

4.1 Submitting a job:

To submit a job to the RAL archiving system from the Durham data server - or replacing Durham with any of the DiRAC sites - (this command is for a single file submission; see next section for the command for multiple files)

```
fts-transfer-submit \
-s https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer \
-v gsiftp://data.cosma.dur.ac.uk/cosma5/data/anel212a/mag_spectrum_0005.dat \
srm://srm-dirac.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk/DiRAC↩
/tape/durham.ac.uk/cosma5/data/dp010/anel212a/mag_spectrum_0005.dat
```

Please note: although there are breaks in the line foreach of the transfer pairs in the above given example, this is an artifact of this documentation. There can be **NO** breaks in the lines!!

A detailed documentation on FTS3 commands can be found in `http://fts3-service.web.cern.ch/content/clients`. The current document is not exhaustive on all the FTS commands.

The above example shows, that the directory structure on the RAL archiving system to which data from Durham is archived is

`vo.dirac.ac.uk/DiRAC/tape/durham.ac.uk`

For the other DiRAC sites the following directories have been allocated (note that the ownership will be changed as more `dirac`XX accounts become allocated to DiRAC sites, *e.g.* `dirac02` will be the owner of the `le.ac.uk` directory):

dirac01	diracvo	cam.ac.uk/
dirac01	diracvo	cambridge/
dirac01	diracvo	damtp.cam.ac.uk/
dirac01	diracvo	durham.ac.uk/
dirac01	diracvo	ed.ac.uk/
dirac01	diracvo	edinburgh/
dirac01	diracvo	epcc.ed.ac.uk/
dirac01	diracvo	le.ac.uk/
dirac01	diracvo	test/
dirac01	diracvo	user/

The directories are all distinct, so we will likely delete the ones we are not using (such as use `cam.ac.uk` instead of `cambridge`).

In the transfers, Durham has chosen to replicate the toplevel directory structure of the Durham filesystem and the full pathname of a user file on the archiving site. Thus the directory structure and the owner can be identified if/when they they are being retrieved from the archive. The ownership of the files as recorded on the archiving system in RAL are dirac0x.diracvo, where x = *1 - 5* for the different DiRAC sites: the original owner and permissions will NOT be preserved.

The user name dirac0x must also exist on the data transfer system of the local DiRAC site, as the transfer agent will open up threads from RAL on the local site with that user name. The RAL endpoint only knows those IDs!

The file structure of the archive can be browsed using the web address: (figure 2)
https://lcgcadm04.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk/DiRAC/tape
from a browser that has a Grid User certificate installed that is enrolled in the DiRAC VO to authenticate. However, as of the end of October 2015, a system is being deployed which will publish

to DiRAC the contents of their directories – this "catalogue dump" is already used by other VOs, and extending the use to DiRAC should be straightforward.

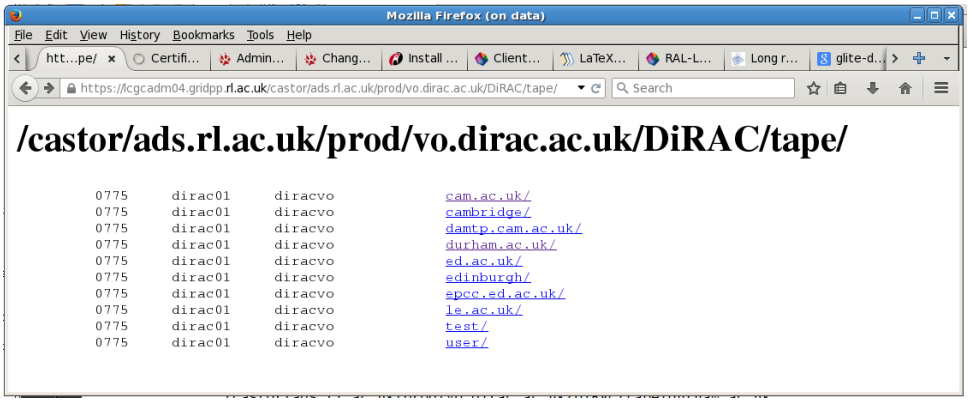

Figure 2: list directories on vo.dirac.ac.uk/DiRAC/tape archive using the web address https://lcgcadm04.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk/DiRAC/tape/

4.2 To submit a job with n files one would create a submission file containing lines of the form

```
gsiftp://data.cosma.dur.ac.uk/cosma5/data/anel212a/mag_spectrum_0001.dat ↩
srm://srm-dirac.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk↩
/DiRAC/tape/durham.ac.uk/cosma5/data/dp010/anel212a/mag_spectrum_0001.dat
gsiftp://data.cosma.dur.ac.uk/cosma5/data/anel212a/mag_spectrum_0002.dat ↩
srm://srm-dirac.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk↩
/DiRAC/tape/durham.ac.uk/cosma5/data/dp010/anel212a/mag_spectrum_0002.dat
....
```

(please note: ↩ indicate that there is NO line break!)

This job can be submitted using the command

```
fts-transfer-submit -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer -v  --retry 3 \
-f name-of-that-file
```

Or writing a script:

```
#!/bin/bash
export X509_USER_CERT=$HOME/.globus/usercert-robot.pem
export X509_USER_KEY=$HOME/.globus/userkey-robot.pem

if [[ $# == 0 ]]; then
   echo "usage: submit submitfile"
   exit 1
fi
file  $1

a="T"

while [[ $a == "T" ]];
do
```

```
file $1
fts-transfer-submit -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer  --retry 2 -e 17280000 \
--timeout 10800 -f $1

if [ $? != 0 ]; then

echo "this did not work; needs to be re-submitted"

else
      a="F"
fi
done
```

In this script a specific user certificate is used. The same user certificate is also used to create a grid proxy. For more on proxies please see the next section 4.3. If the variables X509_USER_CERT and X509_USER_KEY are not specified the files $HOME/.globus/usercert.pem and $HOME/.globus/userkey.pem are expected to exist and will be used.

This script also caters for failed submissions. If the submissions fails - which it might do with an error message like

```
 glite_delegation_delegate: SOAP fault: SOAP-ENV:Client - CGSI-gSOAP
running on data reports could not open connection to lcgfts3.gridpp.rl.ac.uk:8443
 (getaddrinfo failed in tcp_connect())
```

then the script will attempt to submit the job again until it succeeds or until the number of retries (2 in this case) is exhausted.

The default timeout of an fts job is 3600 seconds. If heavy traffic is encountered on the link or if the file is very large, this might not be sufficient. So it is important to increase this. In this example the timeout is 10800 (or 3 hours). In Durham we now use a default 36000 (10 hours) as we are at times in competition of heavy GridPP FTS traffic.

4.3 Creating a proxy

Before an archiving job can successfully be submitted a proxy has to be created and a delegation has to be made to the FTS server. This can be done in two different ways, which are being described in this section.

I. Doing it the grid-proxy way
   This would have no reference to the VO vo.dirac.ac.uk, but it will work and extensive proxy times can be set.

   a. Create the proxy:

      ```
      grid-proxy-init -hours 4800
      Your identity: /C=UK/O=eScience/OU=Durham/L=eScience/CN=lydia heck/CN=Robot:GridClient
      Creating proxy ........................................... Done
      Your proxy is valid until: Thu May  5 15:10:43 2016
      Remaining time for the local proxy is: 4800hours and 0 minutes.
      ```

      This will create a proxy with a life span of 300 days or the life of your current certificate, which ever is shorter, creating a file in /tmp with a name like `x509up_uwxyz` where wxyz is the UID of the user creating the proxy. During the submission process this proxy is required in the authorization process with the archiving server.

   b. Creating a delegation:

```
fts-delegation-init -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/glite-data-transfer-fts/services/FileTransfer \
-e 17200000 -v \newline

fts-delegation-init -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/glite-data-transfer-fts/services/gridsite-delegation \
-e 17200000 -v
```

```
No proxy found on server. Requesting standard delegation.
Requesting delegated proxy for: 4777hours and 46 minutes.
Credential has been successfully delegated to the service.
Remaining time for the local proxy is: 4799hours and 59 minutes.
Remaining time for the proxy on the server side is: 4777hours and 45 minutes.
Not bothering to do delegation, since the server already has a delegated
credential for this user lasting longer than 4 hours.
```

The process identifies that there was no proxy found on the server. Then a delegated proxy is being created for the stipulated time of 17200000 seconds. The next delegation is actually superfluous as there is already a delegation, the former being for the FileTransfer and the second for the gridsite-delegation.

II. Doing it the voms-proxy-init way
This should allocate the files that have been archived to the specific vo of DiRAC: `vo.dir.ac.uk`.
. Create the proxy using the command

```
voms-proxy-init -voms vo.dirac.ac.uk -valid 24:00
```

```
bash-4.1$ voms-proxy-init --voms vo.dirac.ac.uk -valid 24:00
Contacting voms.gridpp.ac.uk:15511
[/C=UK/O=eScience/OU=Manchester/L=HEP/CN=voms.gridpp.ac.uk] "vo.dirac.ac.uk"...
Remote VOMS server contacted succesfully.

creating a file /tmp/x509up_uwxyz

Created proxy in /tmp/x509up_uwxyz.

Your proxy is valid until Wed Aug 05 15:50:44 BST 2015
```

Then created a delegated proxy on the server

```
fts-delegation-init -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/glite-data-transfer-fts/services/FileTransfer \
-e 604400 -v

fts-delegation-init -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/glite-data-transfer-fts/services/gridsite-delegation \
-e 604400 -v
```

Once that is done, submit your jobs.

III. submit a job using

```
fts-transfer-submit -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer -v  -e 604800 --retry 3 \
-f file-list
```

5 Monitoring

5.1 One link that is most useful is the web access to the ganglia output for the archive servers: (please note: any line breaks are artificial and due to the text in latex not wrapping).

```
http://ganglia.gridpp.rl.ac.uk/ganglia/?r=hour&cs=&ce=&c=Storage_CASTOR_Gen&h=&tab=
m&vn=&hide-hf=false&m=network_report&sh=1&z=small&hc=4&host_regex=gdss6%2818
[ganglia.gridpp.rl.ac.uk]|19|20|21|22|23|24%29&max_graphs=0&s=by+name
```

This will help when setting up, as it can give an indication if and at what transfer rate data are being moved, but is of course not unique as other data movement and network activities will overlap with the DiRAC data transfers.

5.2 FTS tools
More immediate monitoring tools are available from the FTS range of commands: `fts-transfer-list` and `fts-transfer-status`.

When a job is submitted it will be allocated a unique identifier of the form `a98849a6-04fd-446d-ae33-39599442fe0e` and a job can queried using it id.

As the command line for each on of those queries can be rather lengthy and as there might be several jobs to query I have created some scripts to check on existing transfers:

check:

```
#!/bin/bash
export X509_USER_CERT=$HOME/.globus/usercert-robot.pem
export X509_USER_KEY=$HOME/.globus/userkey-robot.pem

for i in ‘fts-transfer-list -s https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer \
| grep Request | awk ’{ print $3 }’‘; \
do  fts-transfer-status -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer -v ${i};
done
```

This script would use the correct user certificate and key. Then it would list the current file transfers active or queued assigned to that certifcate and give an output similar to:

```
# Using endpoint : https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer
# Service version : 3.7.6-1
# Interface version : 3.7.0
# Schema version : 3.5.0
# Service features : glite-data-fts-service-3.7.6-1
# Client version : 3.2.32
# Client interface version : 3.2.32
Request ID: a98849a6-04fd-446d-ae33-39599442fe0e
Status: SUBMITTED
Client DN: /C=UK/O=eScience/OU=Durham/L=eScience/CN=lydia heck/CN=Robot:GridClient
Reason: <None>
Submission time: 2015-07-27 13:10:43
Files: 2048
Priority: 3
VOName: nil
        Active: 0
```

```
        Ready: 0
        Canceled: 0
        Finished: 0
        Submitted: 2048
        Failed: 0
        Staging: 0
        Started: 0
        Delete: 0

# Using endpoint : https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer
# Service version : 3.7.6-1
# Interface version : 3.7.0
# Schema version : 3.5.0
# Service features : glite-data-fts-service-3.7.6-1
# Client version : 3.2.32
# Client interface version : 3.2.32
Request ID: f3e4cdb7-f18d-4bf6-b997-9fa365870304
Status: ACTIVE
Client DN: /C=UK/O=eScience/OU=Durham/L=eScience/CN=lydia heck/CN=Robot:GridClient
Reason: <None>
Submission time: 2015-07-27 13:05:43
Files: 2048
Priority: 3
VOName: nil
        Active: 2
        Ready: 0
        Canceled: 0
        Finished: 20
        Submitted: 1125
        Failed: 901
        Staging: 0
        Started: 0
        Delete: 0
```

for more information on fts-transfer-status please check section 3.1 in
`http://fts3-service.web.cern.ch/content/clients`

5.3 Usage figures

The following command will show usage figures: (Please note: the break in the just after "gridpp.rl.ac.uk," is artificial).

```
ldapsearch -x -H
ldap://lcgbdii01.gridpp.rl.ac.uk:2170/[lcgbdii01.gridpp.rl.ac.uk:2170] -s base \
-bGlueSALocalID=DiRACTape,GlueSEUniqueid=srm-dirac.gridpp.rl.ac.uk,
mds-vo-name=ral-lcg2,mds-vo-name=local,o=grid
```

with an output of something like

```
# extended LDIF
#
# LDAPv3
# base
```

```
# <GlueSALocalID=DiRACTape,GlueSEUniqueid=srm-dirac.gridpp.rl.ac.uk
# ,mds-vo-name=ral-lcg2,mds-vo-name=local,o=grid>
# with scope baseObject
# filter: (objectclass=*)
# requesting: ALL
#

# diracTape, srm-dirac.gridpp.rl.ac.uk, RAL-LCG2, local, grid
dn: GlueSALocalID=diracTape,GlueSEUniqueID=srm-dirac.gridpp.rl.ac.uk,Mds-Vo-na
 me=RAL-LCG2,Mds-Vo-name=local,o=grid
GlueSATotalNearlineSize: 0
GlueSAPolicyMaxPinDuration: 0
GlueSAPolicyMaxData: 0
GlueChunkKey: GlueSEUniqueID=srm-dirac.gridpp.rl.ac.uk
GlueSAUsedNearlineSize: 0
GlueSAName: diracTape
GlueSAFreeNearlineSize: 0
GlueSAAccessControlBaseRule: vo.dirac.ac.uk
GlueSAAccessControlBaseRule: VO:vo.dirac.ac.uk
GlueSAReservedNearlineSize: 0
GlueSAExpirationMode: neverExpire
GlueSAPolicyMaxFileSize: 0
GlueSAReservedOnlineSize: 19971
GlueSAAccessLatency: online
GlueSAPolicyQuota: 0
GlueSchemaVersionMajor: 1
GlueSALocalID: diracTape
GlueSAPolicyFileLifeTime: 0
GlueSAPath: /castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk/DiRAC/tape
GlueSARetentionPolicy: replica
GlueSchemaVersionMinor: 3
GlueSAPolicyMaxNumFiles: 0
objectClass: GlueSATop
objectClass: GlueSA
objectClass: GlueSAPolicy
objectClass: GlueSAState
objectClass: GlueSAAccessControlBase
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueSAType: permanent
GlueSATotalOnlineSize: 19971
GlueSACapability: InstalledOnlineCapacity=19971
GlueSACapability: InstalledNearlineCapacity=0
GlueSAPolicyMinFileSize: 200000000
GlueSAUsedOnlineSize: 14977
GlueSAFreeOnlineSize: 4993
GlueSAStateUsedSpace: 14977455434
GlueSAStateAvailableSpace: 4993924584

# search result
search: 2
result: 0 Success
```

```
# numResponses: 2
# numEntries: 1
```

5.4 Monitoring and perusing using web pages:
For a smooth access of any of the monitoring web pages listed below it is required to load the UK eScience CA certificates in the browser They're available http://www.ngs.ac.uk/ukca/certificates/cacerts, see figure 3. The 'UK Root CA' Certificate should be sufficient. Just click on 'UK Root CA' in the table 'IGTF-accredited UK eScience CA Certificates' to load the certificate. No other actions are required!

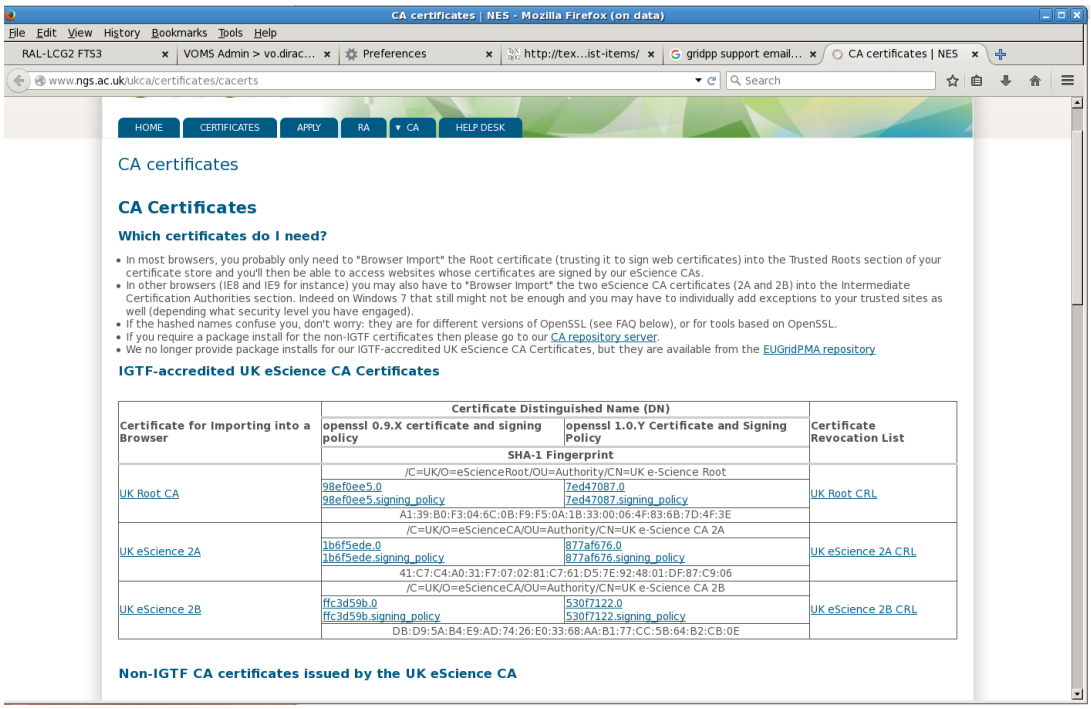

Figure 3: Obtaining the IGTF-accredited UK eScience CA Certificates http://www.ngs.ac.uk/ukca/certificates/cacerts

The web page - which applies to Durham, and will have a different path for the other DiRAC sites

`https://lcgcadm04.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk/DiRAC/tape/durham.ac.uk`

will let you browse those files that have been transfered and will look something like

`/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk/DiRAC/tape/durham.ac.uk/`

```
        0775     dirac01     diracvo             cosma5/
```

and you could click on cosma5 and delve down deeper.

Again only users with a Grid certficate registered with the DiRAC VO can access this site. This is true for any of the web pages that are shown the the reminder of the is document. More than the durham.ac.uk directory are available as you can see from figure 4

5.5 Another site to browse and check on current and recent transactions is - Figure 5

`https://lcgfts3.gridpp.rl.ac.uk:8449/fts3/ftsmon/#/`

Figure 4: list directories on vo.dirac.ac.uk/DiRAC/tape archive using the web address https://lcgcadm04.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk/DiRAC/tape/

The site will offer 4 dialog boxes at the top of the page and you should choose as destination: srm://srm-dirac.gridpp.rl.ac.uk and source gsiftp://data.cosma.dur.ac.uk in the second box; -All- in the first box and the default 1 hour in the last box. Then hit apply and you will get something like shown in figure 5

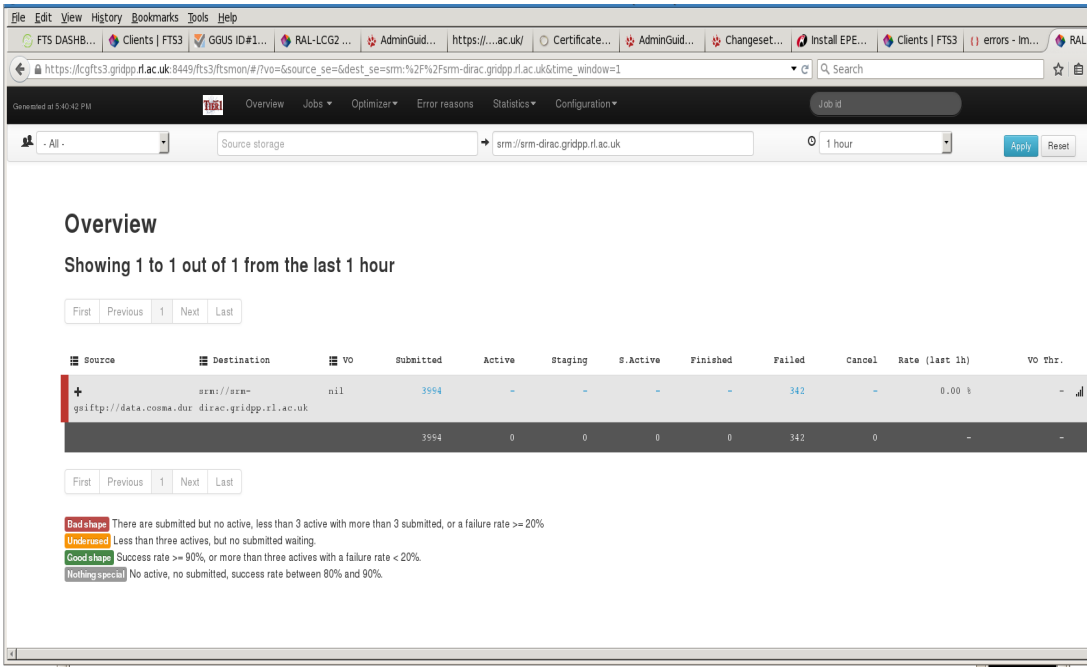

Figure 5: Choose srm-dirac.gridpp.rl.ac.uk and gsiftp://data.cosma.dur.ac.uk

If you then click on the highlighted number under in the column 'finished' on the page shown in in Figure 5 you can see see list of jobs such as in figure 6

Going back to the original figure 5 and click on the highlighted number in the column submitted you will see something like shown in figure 7

Then clicking on any of the strings under the heading Job id, you would see, figure 8 a list of

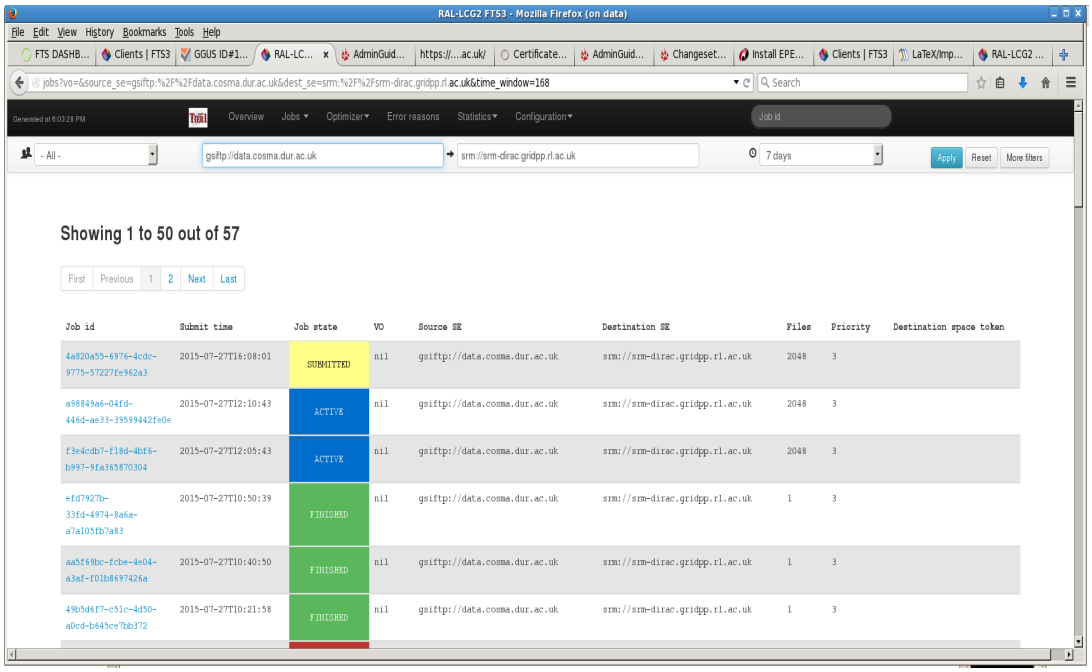

Figure 6: Transactions for those last 7 days from Durham to srm-dirac.gripp.rl.ac.uk

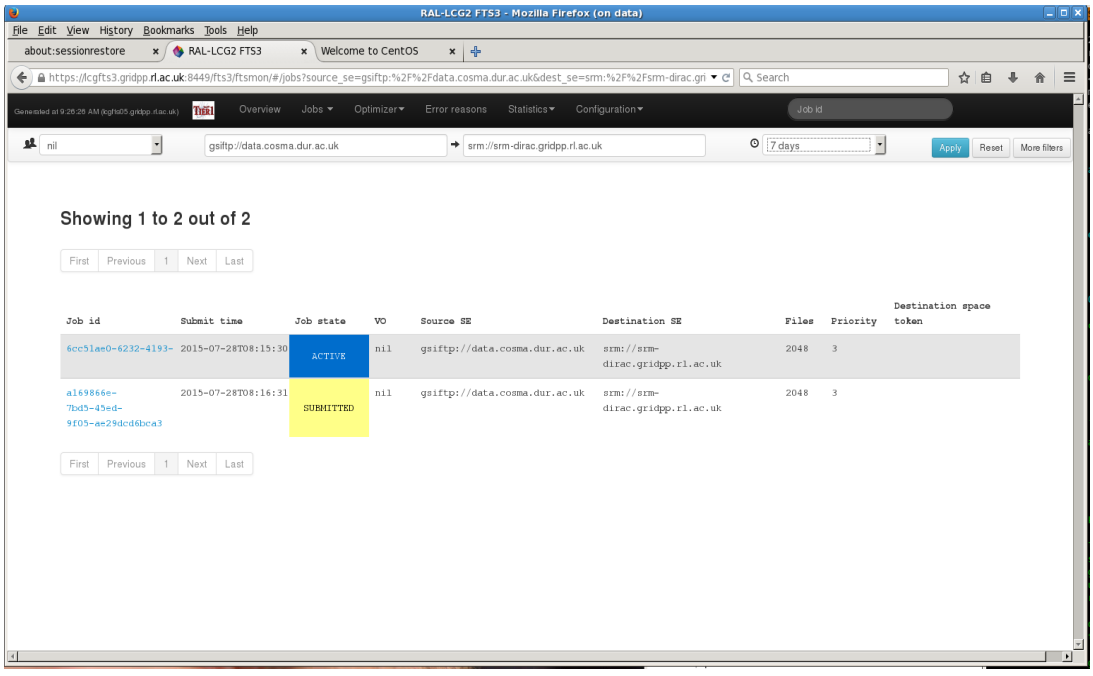

Figure 7: Jobs that are submitted and currently active

submitted files, their state like FINISHED (good thing), submitted (still waiting to be processes), failed - not so good, but the reason can be found out by clicking on the '+' sign at the beginning of that row.

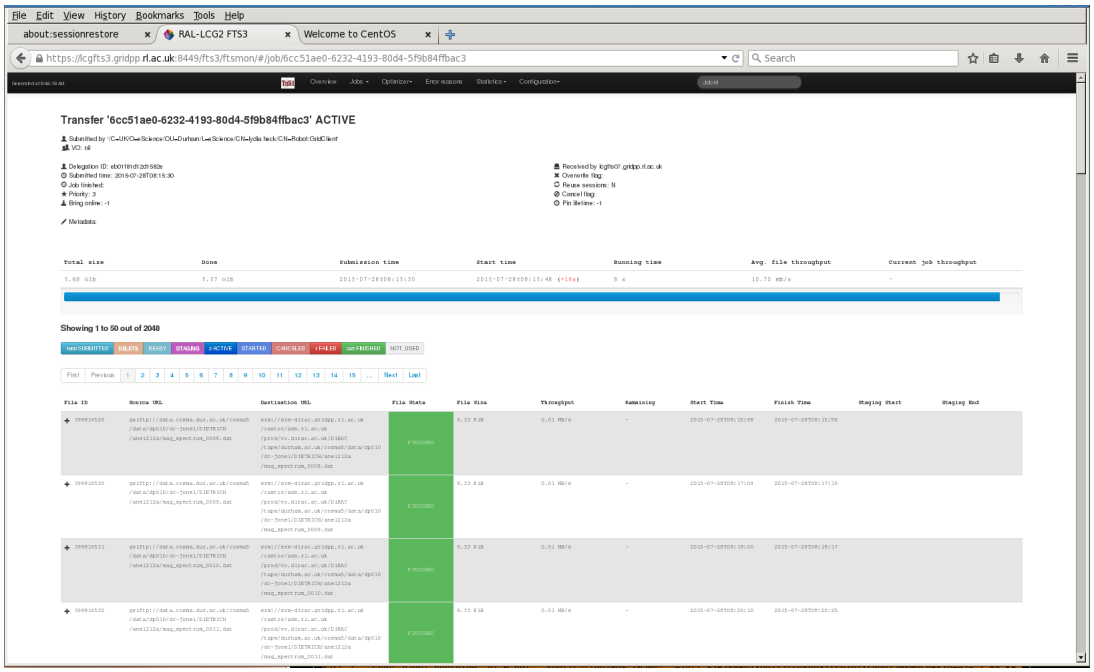

Figure 8: Individual sub-jobs - files to be processed

And an individual file transaction can be queried by clicking on the id on the left of the list with a result as given in figure 9:

Because of the idiosyncracies of users and their file management, it has become clear that just transfering sets of files will make the archiving task impossible. In Durham we have currently close to 70 million files. Some of these files are zero length; some a few bytes big. Directories such as .svn or .csv or .git should not be transfered as the general repositories live somewhere else and they are hopefully backed up and can be retrieved when/if required.

The other drawback of copying each individual file is that ownership, permissions and timestamps are not preserved.

To tackle all these issues, and in discussion with the RAL team, I have written a script that allows for firstly tar'ing a set of no more than 100,000 files with an accumulated size of a given upper limit, whatever comes first. I could only have chosen an upper limit, but initially I used a list of files with *tar*. So I hit limit on the allowed argument length, until I discovered the *-T* option. This option allows to have a list of files in a file itself and there is no argument length issue.

The *tar* is done on one of the fileservers itself. Once the *tar* is completed an *ssh* into the user account with a valid grid certificate on the data transfer system will submit the actual transfer using the *fts-transfer-submit* command. I have written a script to handle submission failures. The script will record the submission id in a file. As this is a batch job submission the command will immediately return.

Due to the space limitation, the new tar files cannot accumulate as eventually we might encroach on valuable research data space. So the script for the transfer submission is set to sleep until the transfer is complete or has failed. This uses the command *fts-transfer-status* on the the submission id. Once the

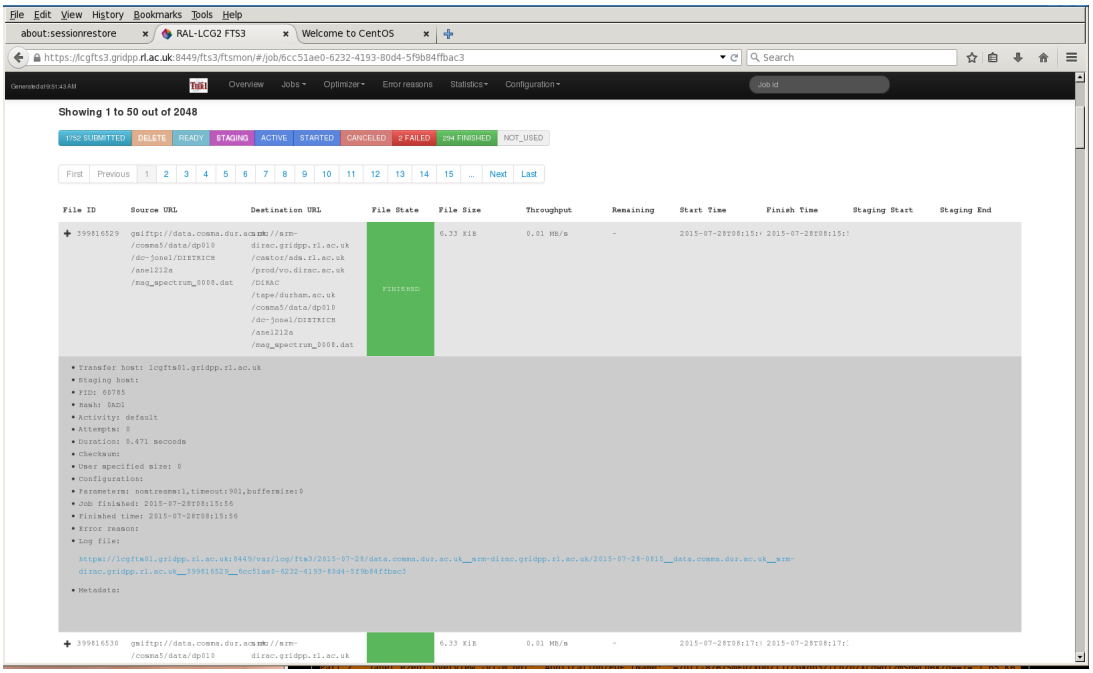

Figure 9: Detailed information on a specific file transaction

transfer is complete or has failed the completion or failure is recorded in the same file that contains the submission id. The script then returns and on the fileserver the tar file will be deleted if the transfer was completed. If the transfer fails the tar file will remain.

Then the next tar file will be prepared.

I have chosen to do the transfers by DiRAC project directories taring up individual user directories within.

In preparation for the tar'ing process, for each of the files its size its full path relative to the project directory is recorded using the *find* command writing the records using a *-printf* instruction and separating the two columns with a ':::' straight in front of the file name. I also use the find instructions to exclude .svn and .git directories and tar files in the user's directory can also be excluded. This is optional and the instruction can be given on the command line. If this option is chosen, the tar files are recorded and archived separately but of course not deleted on a successful transfer.

The things I observed are:

users do use white space (and other weird characters) in their directory and file names. With the above method I hope to have covered all eventualities.

The script is:

```
#!/bin/bash

# Author: Lydia Heck, ICC, Durham
# version 1.4.3
#
# Check if the correct arguments are given to the script:
# 1. argument: DiRAC project directory name
```

```
# 2. should the tar files with a user directory be treated separately

if [[ $# -lt 2 ]]; then
   echo "usage archive-to-ral project do_tar(yes/no)"
   exit
fi


# Set the limit of the size of the tar file.
file_size=274877906944

# Check if the directory for the project exists. This will have to be
# changed for each site. The naming reflects COSMA5's filesystem structure.

if [ -d /cosma5/data/$1 ]; then
   cd /cosma5/data/$1
else
   echo "/cosma5/data/$1 does not exist"
   exit
fi

do_tar=$2

# Run through all user directories within the project:
for i in `ls`; do
#for i in dc-kond1 ; do
    echo ${i}
    if [[ -d $i ]]; then
       stump=${i}_list
# check first if specific recording files already exist.
# If the do, skip this part.
       if [[ ! -f ${stump} && ! -f ${stump}_nr && ! -f ${stump}_na ]]; then
# if these files do not exist, then find all the files for a specific
# user directory excluding
          if [[ "${do_tar}" = "no" ]]; then
             find ${i} -type f ! -path "*/.svn/*" ! -path "*/.git/*"  \
             -printf "%s :::%p\n"  > $stump
          else
             find ${i} -type f ! -path "*/.svn/*" ! -path "*/.git/*" \
             ! -path "*.tar" -printf "%s :::%p\n"  > $stump
          fi
# prepare the file - list: ${stump}_na and file - size: ${stump}_nr
          awk -F":::" '{ print $1 }' $stump > ${stump}_nr
          awk -F":::" '{ print $2 }' $stump > ${stump}_na
       fi

# Are tar files within the user directory to be treated separately?

       if [[ "${do_tar}" = "yes" ]]; then
          if [[ ! -f ${stump}_tar ]]; then
             find ${i} -type f -path "*.tar" -printf "%p\n" > ${stump}_tar
          fi
          if [[ -f ${stump}_tar ]]; then
```

```
            for k in `cat ${stump}_tar`; do
               the_archive=$k
               log_archive=`echo $the_archive | sed -e "s/\//_/g"`
               log_archive=${log_archive}.log
# ssh into the data transfer system as the user,
# who holds the Grid certificate and against whose
# id the Grid dirac account is registered, and submit the transfer job
               ssh dph0elh@data "~dph0elh/GSI/bin/single-file-archive-tar \
               $1 ${k} ${log_archive}"

# When the transfer script returns with completed, ssh back into
# the account and update the log
# archive the log file to completed.a and then remove the log file.
# This could all be done on the file server space, but currently it
# is done in the user space.
# If not completed, do not archive the file and do NOT delete it
     if [ `ssh data.cosma "grep completed ~dph0elh/GSI/LOGs/${log_archive}"` ]; then
        echo "completed"
        echo "This file is part of the structure and will not be deleted"
        ssh dph0elh@data.cosma \
        "cd ~dph0elh/GSI/LOGs; ar cr completed.a ${log_archive}; rm ${log_archive}"
                else
                     echo "not completed"
                     echo "I will now continue with the next"
              fi
            done
          fi
       fi

# Now treat all the user files:
       nlines=`wc -l ${stump}_nr | awk '{ print $1 }'`
       echo "number of lines: $nlines"

# if there is a file resume_${i} ${i}=user directory then identify where to start
# in the file list.

       if [ -f resume_$i ]; then
          first_line=`grep "new_first_line" resume_$i | awk '{ print $2 }'`
          numtar=`grep "last_number_of_tars" resume_$i | awk '{ print $2 }'`
          numtar=$((numtar+1))
       else
          first_line=1
          numtar=1
       fi
       nbytes=0
       n=0
# Run to the file containing the size until the total amount is
# greater or equal to $file_size of the number
# of files is greater or equal to 100000 and add the paths
# to the appropriate files to ${stump}_na_temp
# the start the tar of all those files using the -T flag to read from ${stump}_na_temp.
       for nsize in `tail -n +$first_line ${stump}_nr`; do
           last_line=$((first_line+n))
```

```
           n=$((n+1))

           nbytes=$((nbytes+nsize))
if [[ $nbytes -ge $file_size  ||  $last_line -ge $nlines || $n -ge 100000 ]]; then
               nend=$n
               echo "total number of lines: $nlines"
               echo "number of lines : $nend"

               tail -n +${first_line} ${stump}_na | head -n $nend > ${stump}_na_temp
               tar -cf ${stump}_${numtar}.tar -T ${stump}_na_temp
               first_line=$((first_line+nend))
               echo "new_first_line ${first_line}" > resume_${i}
               echo "last_number_of_tars $numtar" >> resume_${i}
               n=0
               nbytes=0

               log_archive=${stump}_${numtar}.tar.log
   ssh dph0elh@data "~dph0elh/GSI/bin/single-file-archive $1 ${stump}_${numtar}.tar"
   if [ `ssh data.cosma "grep completed ~dph0elh/GSI/LOGs/${log_archive}"` ]; then
                  echo "completed"
ssh dph0elh@data.cosma "cd ~dph0elh/GSI/LOGs; ar cr completed.a ${log_archive}; rm ${log_archive}"
                  echo "I should now delete the tar file ${stump}_${numtar}.tar"
                  /bin/rm ${stump}_${numtar}.tar
                  # /bin/rm
               else
                  echo "not completed"
                  echo "I will now continue with the next"
               fi

               echo "do you want to continue?"
               #read y
               y="y"
               if [ "$y" = "n" ]; then
                  echo "done"
                  exit
               fi
               numtar=$((numtar+1))
           fi
       done
    fi
done
```

The script single-file-archive is now at is version 1.2 and looks as follows

```
#! /bin/bash  -x
# author: EL Heck, ICC, Durham University
# February 2016
# load a specific certificate. - I have two.
export X509_USER_CERT=$HOME/.globus/usercert-robot.pem
export X509_USER_KEY=$HOME/.globus/userkey-robot.pem

if [[ $# == 0 ]]; then
```

```
   echo "usage: single-file-archive project tarfile"
   exit 1
fi

# Enter log directory
cd $HOME/GSI/LOGs

# Define location of tar file and tar file variables.
the_dir=/cosma5/data/$1
the_archive=$2

# Sometimes a submission fails because one of the many FTS servers fails to respond.
# To catch this, the script will loop until the file is submitted.
a="T"
while [[ $a == "T" ]]; do

# The submission will be tried 3 times, before a failed would be reported:
# --retry 2
# The proxy should live for 200 days or until the certificate expires,
# whatever comes first: -e 17280000
# The archive of this session can last up to 10 hours, before it would be
# killed. If it was kill because
# of time-out it would fail: --timeout 36000
# The next argument is the path to the file location that is about to be
# transfered in gsiftp syntax
# The file argument is the location and path of the archived file.
# The session is given a submission identifier, which will be listed in the log file.

fts-transfer-submit -s \
https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer  -o --retry 2 \
-e 17280000 --timeout 36000 \
gsiftp://data.cosma.dur.ac.uk${the_dir}/${the_archive}
srm://srm-dirac.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/vo.dirac.ac.uk/DiRAC\\
/tape/durham.ac.uk${the_dir}/${the_archive} > ${the_archive}.log
# PLEASE NOTE: the break in the last two lines is artificial and due
# to lines not wrapping in verbatim mode.
if [ $? != 0 ]; then

                echo "this did not work; needs to be re-submitted"

                else
                a="F"
        fi
done

# The script now goes into wait statement until the transfer is
# complete or has failed.
# This will be recorded in the log file.
contd=true
ID=‘cat ${the_archive}.log‘
while [ "$contd" = true ]; do
sleep 120
```

```
# The status is checked using the command fts-transfer-status.

subm_status=‘fts-transfer-status \
-s https://lcgfts3.gridpp.rl.ac.uk:8443/services/FileTransfer \
-v $ID | grep Status | awk ’{ print $2}’‘
if [[ "$subm_status" = "FINISHED" ]]; then
   echo "completed" >> ${the_archive}.log
   contd=false
else
   if [[ "$subm_status" = "FAILED" ]]; then
      echo "failed" >> ${the_archive}.log
      contd=false
   fi
fi
done
```