
Grid Documentation Documentation

Release 1.0

Grid Support <helpdesk@surfsara.nl>

Apr 09, 2024

CONTENTS

1	General	3
2	Basics	9
3	Advanced topics	23
4	Best practices	89
5	Service implementation	107
6	Tutorials	111
7	Frequently asked questions	119
8	Indices and tables	131

Welcome to the documentation for using the Grid services at [SURFsara](#). The information in this tutorial will help you get started with the Grid, learn best techniques to successfully port your application to the Grid infrastructure and stay up-to-date with our system developments. We welcome your comments at helpdesk@surfsara.nl or your *contribution* to help us improve the documentation.

Need help?

Do you need help with this tutorial? We are more than willing to assist! Just contact us at helpdesk@surfsara.nl.

1.1 About the Grid

In this page you will find some general information about the Grid, how it works and what is suited for:

Contents

- *About the Grid*
 - *Introduction to Grid*
 - *To use the Grid or not*
 - *How it works*

1.1.1 Introduction to Grid

The Grid is a loosely coupled collection of heterogeneous resources with a large variety in purpose and reliability. The Grid consists of multiple geographically distributed compute clusters interconnected with fast network. Grid computing is different from conventional high performance computing such as cluster computing in that Grid systems typically have each processor core perform an independent task.

More about Cluster computing basics?

See also:

Check out our mooc video *Cluster Computing*

In addition, Grid clusters are not exclusively connected to their own storage facilities but can be accessed transparently from all Grid compute clusters, and as such form a virtual supercluster. Grid systems and their applications are more scalable than classic clusters.

The Grid is especially suited, therefore, to applications that cannot be solved in a single cluster within a reasonable timeframe. Common examples are the Large Hadron Collider (LHC) experiments, large-scale DNA analyses and Monte-Carlo simulations.

1.1.2 To use the Grid or not

Grid computing is a form of distributed computing which can be *very powerful when applied correctly*. Grid is best suited for applications with a data-parallel nature that require many simultaneous *independent* jobs. With the help of the Grid, large scale computational problems can be solved and large amounts of data can be handled and stored.

Note: The Grid suits applications that can be split up relatively easily in multiple, independent parts or else **embarrassingly parallel** jobs.

Other HPC options

The Grid will be an interesting service if you are faced with workloads that concern hundreds of thousands of core hours and/or many terabytes of data. For other applications that concern small data or compute requirements, please have a look for other suitable [HPC systems](#) at SURFsara.

Job submission has a relatively high overhead. Submitting a “hello world” program may take minutes. Your data and your software must be available on the worker nodes, which requires careful planning of the job workflow. With the size of the job collections typical for the Grid, and submitting hundreds or even thousands jobs simultaneously, it may become a challenge to check your jobs for status and reschedule based on judgement of failures and their causes. We offer tools to help you automate these actions (see [Pilot jobs](#)), however, porting of your solution to the Grid will always require time and effort to set up. Our experienced consultants are available for assistance and to help you make the right decisions right from the beginning.

The Grid infrastructure is able to accommodate a variety of communities and scientific fields, each with their own type of application and requirements, and without mutual interference. Typical Grid applications are:

- Massive data processing workloads.
- Large computational job collections that require a minimal time to completion.
- Projects that require collaboration and resource sharing with national or international partners.

1.1.3 How it works

As a user you connect to the Grid by connecting to a so-called **User Interface** (UI) system via secure shell. Once you have received the right credentials for a UI (User Interface) (see [Preparation](#)) you are set to go.

In general your task needs to be split into smaller units, called **jobs**, that each fit a certain set of boundary conditions in terms of resources (typically runtime, memory, disk size). For the jobs to be executed on the Grid, a job slot needs to be selected based on the boundary conditions that suit the requirements for these jobs. The way to do this is to describe each job in terms of a Job Description Language (JDL), where you list which program should be executed and the requirements of the job slot to run the job. You can use the **input and output sandboxes** to send small data files or scripts with your job.

More about Grid basics?

See also:

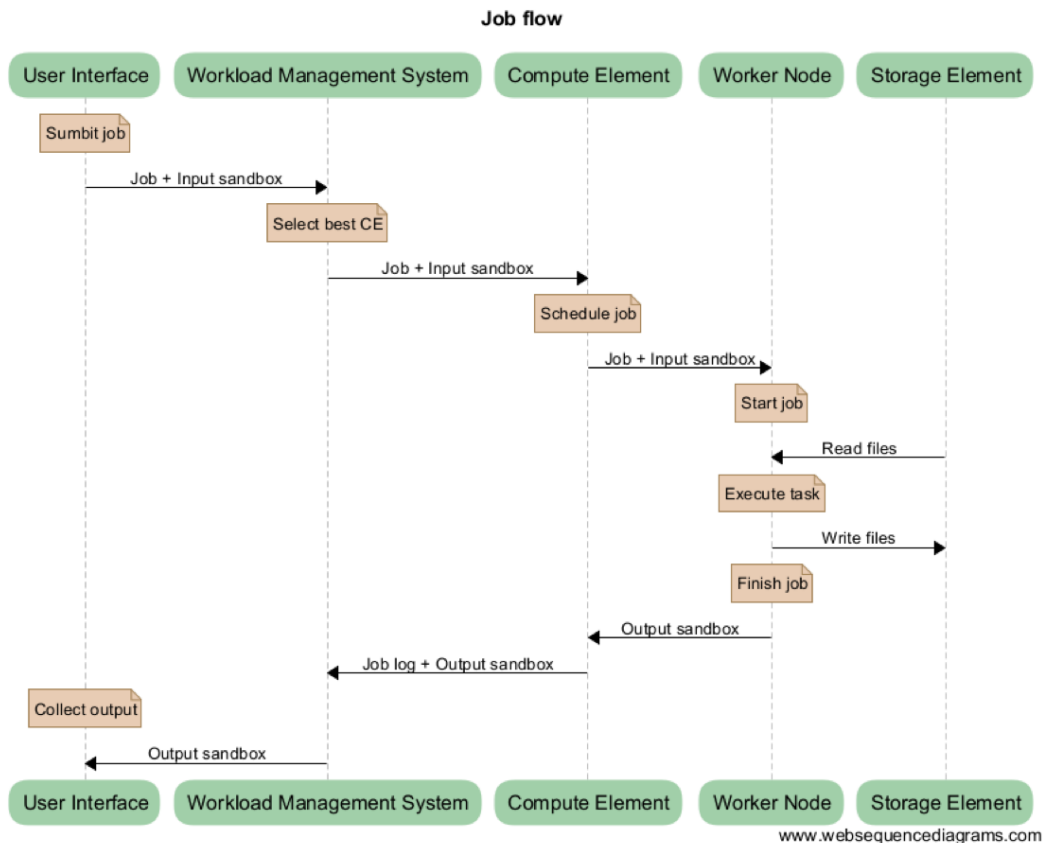
Check out our mooc video [Grid Computing Overview](#)

Each job is then submitted as a JDL (Job Description Language) file to the **Workload Management System** (WMS). The WMS is a *resource broker* that knows which Grid compute clusters are ready to accept your job and fulfil its requirements. Each Grid cluster consists of a **Compute Element** (CE) and several **Worker Nodes** (WNs). The CE

(Compute Element) is a scheduler within each Grid cluster that communicates with the WMS (Workload Management System) about availability of the job slots in the cluster, and accepts and distributes the jobs to the available compute nodes in the cluster (these are called Worker Nodes or WNs). These WNs (Worker Nodes) are the machines which do the actual work. When finished with a job they will report back to the CE, which in turn will inform the WMS about the status of the job.

In addition, the Grid's interconnected clusters each have a storage server, called a **Storage Element** (SE), which can hold the input and output data of the jobs. Data on the SEs (Storage Elements) can be replicated at multiple sites if needed for scale-out scenarios. In general, all SEs offer disk storage for the staging of datasets before and after job execution. In addition, a central Grid storage facility (see dCache) also provides tape storage for long-term storage of datasets that need to be preserved.

In short, as a user you submit your jobs to execute your calculation or analysis code and to handle your input and output data. The WMS distributes the jobs to the clusters and node that are most suitable for these jobs. When the jobs are finished, you can collect the results from the SE (Storage Element) that was selected to hold the output data or keep them for later use on the central Grid storage facility.



1.2 Dutch National Grid

In this page you will find information about the Dutch National Grid Infrastructure:

Contents

- *Dutch National Grid*
 - *About*
 - *National and European*

1.2.1 About

The Grid in general consists of a large number and variety of clusters which are distributed all over the Netherlands and abroad. Grid infrastructures in European member states are organised by means of their own **National Grid Initiatives** (NGIs). SURFsara offers Grid services in collaboration with [Nikhef](#) and [RUG-CIT](#), and together we form the **Dutch National Grid Initiative** (NGI_NL).

The Dutch National Grid Initiative provides the Grid infrastructure to scientists affiliated with Dutch universities and other research institutes, and to international research projects that have established agreements with us.

Grid technology can be used for a variety of purposes and application types, however a subset of the worldwide Grid infrastructure can be dedicated for a specific purpose. For example our [Dutch Grid](#) infrastructure, which consists of a small number of tightly interconnected clusters, operated under the umbrella of NGI_NL, is very potent in enabling *high-throughput processing of large datasets* in a minimum amount of time.

1.2.2 National and European

The Dutch National Grid Infrastructure is connected to [EGI](#), the European Grid Initiative, which allows Grid users in the Netherlands and abroad easy access to one another's resources. EGI also supplies central monitoring and security and ensures smooth interoperability of all connected clusters.

1.3 Grid services

In this page you will find general information about the SURFsara Grid services and support:

Contents

- *Grid services*
 - *About*
 - *Support*

1.3.1 About

To deploy your production tasks smoothly on the Grid, our services can be of great value. Our standard Grid services such as the Workload Management System, Compute Elements, Storage Elements, a Logical File Catalog, Virtual Organisation Management Services, and User Interface machines are indispensable for generic use of the Grid infrastructure. Apart from those we have additional services available as an option, that may be beneficial for your type of application.

Our Grid services provide:

- High-throughput processing nodes for job execution
- Access to scalable *Grid storage* facilities, disk and tape
- Standard Grid Services like User Interfaces, Brokers, and Virtual Organisation management services
- *Token Pool Services (Picas Overview)* for production run logistics, job collection management, monitoring and control
- Virtual filesystem service *Softdrive* for centralized software deployment on distributed systems
- Dedicated *light paths* : we offer support bridging the last mile between the end points and your data sources.
- *Consultancy* : advice and support on getting access, working with Grid certificates, basic job submission, data access methods, best practices, advice on design and optimization of applications for performance improvements, integration with large-scale job-submission frameworks, and international upscaling.

1.3.2 Support

You may request support for our Grid services by contacting us by phone or email. Our dedicated team at [SURFsara helpdesk](#) is more than willing to assist you for any questions or complaints and carefully take note of your remarks for further improvement of our services.

Check out the detailed information about [SURFsara helpdesk](#). Please don't hesitate to contact us!

1.4 How to get access

In this page you will find general information about getting access to the National Dutch Grid infrastructure:

Contents

- *How to get access*
 - *Access to the National Dutch Grid*
 - * *Estimate your resource needs*

1.4.1 Access to the National Dutch Grid

Access to the Dutch Grid clusters allows for:

- submitting *Grid jobs* to multiple clusters via the Grid middleware
- storing data to the *Grid storage*

Researchers at SURF-affiliated institutes can apply for compute and storage capacity on the Grid by submitting the [SURFsara application form](#). For scientists not affiliated to SURF, rates are based on tailor made packages. Specific details on obtaining accounts can be found in the [Access Grid](#) section of our website.

Please contact us at helpdesk@surfsara.nl for any inquiry on our possibilities.

Estimate your resource needs

When you request to use the Grid you do so in the context of a project. A project is set to solve a problem, defined as some goals that you want to achieve, which includes a plan on how you will achieve those goals. For each project a suitable amount of resources will be allocated. You can work together with several people in the same project and using the same resource allocation.

In general, each project resource allocation involves:

- an amount of compute time, measured in core-hours
- an amount of storage space, for disk or tape or both
- a start and end date for your project

In order for us to make a suitable allocation for a project, we need to know what your goals are and how you want to achieve those goals. That is why it is important for you to understand how to estimate the resources for your project.

Not sure how to calculate your resource requirements?

Contact us at helpdesk@surfsara.nl and we can work together on estimating the resources for running your computation.

For a proper estimation of resources requirements it is best to start with a few test runs of the code (if existing) on another system (e.g.: your laptop). The outcome of such tests will in many cases be sufficient to derive a total requirement for your larger-scale production runs. Sometimes a more elaborate process is needed to come to an estimate, for example if the code does not exist yet or if changes to the existing code are still pending. Ideally you have been running a few representative samples of your runs before you file a resource request, in order to have some concrete information ready about the resources needed by your jobs, such as:

- how long it takes to run a representative input scenario
- how much space you need to store input, output and intermediate data
- what the software requirements are (required software tools, libraries, compilers, etc.)
- how many scenarios (jobs) you need to run for a complete analysis

In case you don't know how to prepare such an inventory, we would be happy to assist you with that.

2.1 Prerequisites

This section summarises all the preparations you need to make before you run a simple job on the Grid:

Contents

- *Prerequisites*
 - *Preparation*
 - *Get a User Interface account*
 - *Get a Grid certificate*
 - *Join a Virtual Organisation*

2.1.1 Preparation

The Grid is a cooperation of many different clusters and research organisations, and as such, there is no centralised user management. Yet, there must be a way for the system to identify you and your work. This is why **Grid certificates** and **Virtual Organisations** (VOs) are introduced.

More about Grid prerequisites?

See also:

Check out our mooc video [Working Environment - Grid prerequisites](#).

Your digital identity starts with a private key. **Only you** are allowed to know the contents of this key. Next, you need a Grid certificate, which is issued by a Certificate Authority (CA). The Grid certificate contains your name and your organisation, and it says that the person who owns the private key is really the person mentioned, and that this is certified by the Certificate Authority.

Now this is your identity. Big international cooperations do not want to deal with every user individually. Instead, users become part of Virtual Organisations. To give an analogy, the Grid certificate provides authentication (identity, e.g., like a passport) and the VO provides authorisation (approval, e.g., like a visa). Individual clusters give access and compute time to certain VOs, and if you are a member of a VO (Virtual Organisation), you can run your jobs on that cluster.

More about Grid Security?

See also:

Check out our mooc video *Grid Certificate - Security*.

In order to run your work on the Grid, you have to make three essential steps:

1. *Get a User Interface account*, so that you can interact with the Grid.
2. *Get a Grid certificate*, so that you can be identified on the Grid.
3. *Join a Virtual Organisation*, so that you can run your jobs on the Grid.

The UI account will provide you with the proper environment to submit your jobs to the Grid. The Grid certificate is required to authorise you for using the Grid. Finally, the VO membership is based on your research domain (e.g. lsggrid for Life Scientists) and determines which resources you can use.

These steps are described in this chapter.

Warning: Several of the steps below require to be performed on your web browser. We strongly advice you to use **Firefox** because other browsers may give you errors in the certificate procedure.

2.1.2 Get a User Interface account

In this section we will describe how to get a User Interface account.

The User Interface (UI) account will provide you with the environment to interact with the Grid. It is your access point to the Grid. You log in to a UI machine via SSH. On this computer you can, amongst others, do the following things: access Grid resources, create proxies, submit jobs, compile programs or prototype your application. For debugging purposes it is good to know that the environment of a user interface is similar to a Grid worker node thus if your code runs on the user interface it will most likely run on a Grid worker node.

To request for a UI account, please send us your request at helpdesk@surfsara.nl and we will give you an account on the SURFsara Grid UI (server name: `ui.grid.sara.nl`).

Please note that the UI is simply a Linux machine and working on it requires some familiarity with linux commands and remote access methods. If you need help with this, check out our mooc video *Working Environment - Remote access*.

2.1.3 Get a Grid certificate

Grid certificates are supplied by a Certificate Authority (CA). Users affiliated with Dutch institutes can request a digital certificate either by Sectigo or DutchGrid CA.

How to obtain a Grid certificate?

See also:

Find detailed info in our mooc video *Obtain a Grid Certificate*.

If you are a researcher in the Netherlands we recommended you to request a certificate via Sectigo, by using your institutional login. This is the easiest and fastest way to get a Grid certificate. In cases that the Sectigo option is not applicable, you can request a DutchGrid CA certificate.

Here you can find details for obtaining and installing a Grid certificate:

Sectigo certificate

This section describes how to obtain and install a Sectigo Grid certificate. This is a prerequisite to get started on the Grid.

Contents

- *Sectigo certificate*
 - *Obtain a Sectigo certificate*

Obtain a *Sectigo* certificate

Sectigo allows you to get your Grid certificate *instantly* from the GEANT Trusted Certificate Service (former was the Terena portal and then Digicert), by using your institutional login ([SURFconext](#)).

The following guide will help you:

- obtain a Sectigo certificate instantly from a portal
- change the certificate formats
- install a Sectigo certificate on your own computer or UI
- install a Sectigo certificate in your browser

Instructions on acquiring and setting up a Sectigo certificate can be found in the step by step guide below. For Grid, make sure you select the type *GEANT Personal Authentication* certificate as explained in the guide: <https://ca.dutchgrid.nl/tcs/TCS-enduser-request-guide-NL.pdf>

Warning: If you saved the certificate files on your laptop, please make sure that you copy it from your local machine to your `.globus` directory on the UI.

DutchGrid certificate

This section describes how to obtain and install a DutchGrid Grid certificate. This is a prerequisite to get started on the Grid:

Contents

- *DutchGrid certificate*
 - *Obtain a DutchGrid certificate*
 - * *Request a DutchGrid certificate*
 - * *Retrieve your DutchGrid certificate*
 - *Install a DutchGrid certificate on the UI*
 - *Install a DutchGrid certificate in your browser*

- * *Convert PEM to pkcs12*
- * *Import the certificate to the browser*

Obtain a DutchGrid certificate

In case that your institute does not support SURFconext and is not possible to get a *Sectigo certificate*, then you can apply for a DutchGrid CA certificate. You can request a DutchGrid certificate by launching the JGridstart tool.

Request a DutchGrid certificate

Note: If you are on macOS, ensure that `xquartz` is installed with homebrew.

- Log in to your *UI account* with X forward enabled, e.g.:

```
$ssh -Y homer@ui.grid.sara.nl # replace "homer" with your username!
```

In the case of running on macOS, it may be necessary to specify the path of XAuth in `~/.ssh/config` with:

```
XAuthLocation /opt/X11/bin/xauth
```

- Download the the jGridstart tool:

```
$wget https://ca.dutchgrid.nl/start/jgridstart-wrapper-1.18.jar
```

- Run the wizard:

```
$java -jar jgridstart-wrapper-1.18.jar
```

- Follow the wizard instructions. You will typically go through these steps:
 - Start the Wizard by pressing `Request new ..` button
 - Generate request by entering your details (name, surname, email, organisation). At this stage you will provide the password for your Grid certificate - make sure you keep this safe!
 - Submit request. This will create your private `userkey.pem` file in your `~/.globus` directory.
 - Fill in and print the verification form by pressing the `display form` button. Once you fill in the form, save it locally.
 - Close the wizard
- Check your details in the printed form and contact your institution's **Registration Authority (RA)** in person. The RA person will check your identity (id or passport or driving license) and sign the printed form.
- Once your form is signed by the RA (Registration Authority), send a scanned copy to the DutchGrid CA via email or fax. The contact details can be found in the printed form, but you can contact also helpdesk@surfsara.nl if you are in doubt.
- The DutchGrid CA (Certificate Authority) will finally send your certificate via email within ~a week. Once you have a received your certificate you will need to install it both on your *UI account* and your browser (UI or laptop). We'll see this next.

Note: If you need help to obtain your DutchGrid certificate, please read the [JGridstart guide](#) or contact us at helpdesk@surfsara.nl.

Retrieve your DutchGrid certificate

Once your request is approved, you will receive an email titled “*DutchGrid CA certificate ...*”. Now you need to retrieve the new certificate:

- Log in to your [UI account](#) with X forwarding enabled, e.g.:

```
$ssh -Y homer@ui.grid.sara.nl # replace "homer" with your username!
```

- Run the wizard again:

```
$java -jar jgridstart-wrapper-1.18.jar
```

Then a window pops up similar to the following:



- Click on **retrieve your certificate**. This will automatically create a file `usercert.pem` in your `~/globus` directory (check with `$ ls ~/globus`).
- You may skip the step “install in browser” because the X session on the UI is slow and will probably be interrupted. Just click “Next”
- Close the wizard.

If everything went well, your certificate and key files (`usercert.pem` and `userkey.pem`) should be in the `~/ .globus` directory.

Install a DutchGrid certificate on the UI

If you followed the steps above properly, then your DutchGrid certificate and private key file should now be present in the `~/ .globus` directory (notice the dot!) on the User Interface machine. All you need to do is to set the proper permissions.

- Log in to your *UI account*:

```
$ssh homer@ui.grid.sara.nl # replace "homer" with your username!
```

- Set the proper permissions to your certificate files:

```
$cd $HOME/.globus
$chmod 644 usercert.pem
$chmod 400 userkey.pem
```

Note that the private key file should be **read-only** and only readable to you.

- Verify the correct permissions:

```
$ cd $HOME/.globus
$ ls -l
-rw-r--r--      1 homer    homer          4499  May 10 13:47  usercert.pem
-r-----      1 homer    homer           963  May 10 13:43  userkey.pem
```

Install a DutchGrid certificate in your browser

In order to apply for a *VO membership* you will have to install your certificate in your browser. Note that you can do this from any browser, however for convenience we will describe the procedure using the UI browser.

- Log in to your *UI account*:

```
$ssh -Y homer@ui.grid.sara.nl # replace "homer" with your username!
$cd $HOME/.globus
```

Warning: You can import a certificate in your browser only when it is in the **PKCS12** format. This means that you need to convert the `usercert.pem` and `userkey.pem` files to a single `.p12` file.

Convert PEM to pkcs12

- To convert a PEM file to the PKCS12 format, run on the UI:

```
$openssl pkcs12 -export -inkey userkey.pem -in usercert.pem -out browsercert.p12
```

This will ask you for a password three times: the first is to unlock your private key stored in the file `userkey.pem`. The PKCS12-file will be password protected, which needs a new password, and the same password for confirmation. Note that you can use the same password as the password for the private key file, but this is not necessary.

Import the certificate to the browser

- To import the .p12 file in your browser, open a Firefox window (\$ `firefox` &) on the UI and apply the following steps (Note that you may have to copy the .p12 file to a directory accessible from your browser):
 - From the Firefox Menu bar select:
 - * For Firefox versions older than v57.0: `Edit > Preferences > Advanced > View Certificates > Import`
 - * For Firefox versions higher than v57.0: `Firefox > Preferences > Privacy & Security > scroll to the bottom "Security" section > View Certificates > Import`
 - Select the browsercert.p12 file from the UI local directory
 - Give the password you set in the previous step.
 - You should now see the certificate listed. Close the window.

Problems installing the certificate?

See also:

Need more details for installing your certificate on the UI or browser? Check out our mooc video *User Interface machine*.

- Verify that your certificate is valid and properly installed in your browser by accessing this website:

<https://voms.grid.sara.nl:8443/vomses/>

If you receive an SSL authentication error, then try repeating the steps carefully as they come. If you managed to access the page above, your certificate is successfully installed!

See also:

Does my key match the certificate?

What is the expiry date of my certificate?

How can I see the subject of my certificate?

If you're not affiliated with a Dutch institute, please find a suitable CA at <https://www.igtf.net/pmamap> to obtain a Grid certificate from. We also have a list of CAs that are supported on the grid.

2.1.4 Join a Virtual Organisation

More about VOs?

See also:

Need to know more about VOs and how to get a membership? Check out our mooc video *Virtual Organisations*.

A Virtual Organisation or VO is a group of geographically distributed people that have common objectives and that are using shared Grid resources to achieve them. Every Grid user is a member of one or more Virtual Organisations.

In practice your VO membership determines to which resources (compute and storage) you have access to. You are eligible to register for a VO only once you *get a valid certificate*. The VO that is most suitable for you depends on the specific research area you are in. For example, if you are active in a field associated with the life sciences the `lsgrid` VO might be most suitable for you. If you still not sure which VO is best for you, then contact us at helpdesk@surfsara.nl to guide you on this.

This section describes how to get a VO membership.

Warning: At this point you must have the certificate successfully installed in your browser. If you don't have that, go back the *previous step*.

- Open the link below from the browser where your certificate is installed (UI or laptop). The page will ask you to confirm with your installed browser certificate: <https://voms.grid.sara.nl:8443/vomses/>
- Select the VO that you are interested for e.g. `lsgrid`, from the front page listing all the available VOs (Virtual Organisations). If it is unclear to you for which VO you should register, please contact us at helpdesk@surfsara.nl.
- Fill in some of your personal details (name, email, etc.), then read and accept the AUP (Acceptable Use Policy).
- Check that you have received a verification email titled “Your membership request for VO ...” and confirm with the URL, as described in the email.
- You will receive an approval email titled “Your vo membership request for VO `lsgrid` has been approved” when the VO administrator finally signs your request.
- As soon as your VO membership is approved, the final step is to be added to the Dirac VO user group. For this, please contact us at helpdesk@surfsara.nl.

Once you finish this page instructions successfully, you are set to go and run your *First Grid job with Dirac!*

2.2 First Grid job with Dirac

Our Grid facility is powered by the `DIRAC` service. `DIRAC` provides the workload management system to submit and manage user jobs on the Grid. We provide support for the command line tools of `DIRAC` and access to the `DIRAC` clients on our Grid UI.

This section summarises all the steps to submit your first job on the Grid, check its status and retrieve the output:

Contents

- *First Grid job with Dirac*
 - *Grid job lifecycle*
 - *Dirac proxy creation*
 - *Describe your job in a JDL file*
 - *Submit the job to the Grid*
 - *Track the job status*
 - * *Cancel job*
 - *Retrieve the output*
 - * *Check job output*
 - *Recap & Next Steps*

Warning: You can continue with this guide *only after* you have completed the *preparations* for Grid. At this point your certificate should be installed on the UI and your VO membership should be approved and registered to the Dirac user group. Still need help with any of these steps? We can help! Contact us at helpdesk@surfsara.nl.

Once you finish with the *First Grid job with Dirac*, you can continue with more *advanced topics* and also *Best practices*, the section that contains guidelines for porting real complex simulations on the Grid.

2.2.1 Grid job lifecycle

Dirac

See also:

You can find more general details about various functionalities possible with Dirac in their [project documentation](#)

To run your application on the Grid you need to describe its requirements in a specific language called **job description language** (JDL). This is similar to the information that we need to specify when we run jobs using a local batch scheduling system (e.g., with PBS, SLURM), although it is slightly more complex as we are now scheduling jobs across multiple sites.

Except for the application requirements, you also need to specify in the JDL the content of the *input/output sandboxes*. These sandboxes allow you to transfer data to or from the Grid. The input sandbox contains all the files that you want to send with your job to the worker node, like e.g. a script that you want executed. The output sandbox contains all the files that you want to have transferred back to the UI.

Note: The amount of data that you can transfer using the sandboxes is very limited, in the order of a few megabytes (less than **100MB**). This means that you should normally limit the input sandbox to a few script files and the output sandbox to the stderr and stdout files.

Once you have the JDL file ready, you can submit it to multiple clusters with `dirac-*` commands. Dirac will schedule your job on a Grid worker node. The purpose of Dirac is to distribute and manage tasks across computing resources. More specifically, Dirac will accept your job, assign it to the most appropriate Computing Element (CE), record the job status and retrieve the output.

2.2.2 Dirac proxy creation

Before submitting your first Grid job, you need to create a *proxy* from your certificate. This has a short lifetime and prevents you from passing along your personal certificate to the Grid. The job will keep a copy of your proxy and pass it along to the Worker Node.

This section will show you how to create a valid proxy:

- Log in to your UI account:

```
$ssh homer@ui.grid.surfsara.nl # replace "homer" with your username
```

- To enable the software environment to use Dirac tools, please run the following command:

```
$source /etc/diracosrc
```

Please note that you need to run this command every time you login to the UI. You may also add this command in your configuration file (\$HOME/.bashrc).

- Create a proxy with the following command and provide your Grid certificate password when prompted:

```
$dirac-proxy-init -b 2048 -g projectmine.com_user -M --valid 168:00
```

Each VO (e.g., lsgrid in the above example) is mapped to a group in Dirac (lsgrid_user in this case) and may have a different name than the VO itself. Please contact helpdesk@surfsara.nl if you are unsure of the group name to use. The above command creates a local proxy with a validity of maximum 7 days.

You should see a similar output displayed in your terminal:

```
Generating proxy...
Enter Certificate password:
Added VOMS attribute /lsgrid
Uploading proxy..
Proxy generated:
subject      : /DC=org/DC=terena/DC=tcs/C=NL/O=SURF B.V./CN=homer
↪homer@example.com/...
issuer       : /DC=org/DC=terena/DC=tcs/C=NL/O=SURF B.V./CN=homer
↪homer@example.com/...
identity     : /DC=org/DC=terena/DC=tcs/C=NL/O=SURF B.V./CN=homer
↪homer@example.com
timeleft     : 167:53:58
DIRAC group  : lsgrid_user
path        : /tmp/x509up_uxxxx
username     : homer
properties   : NormalUser
VOMS        : True
VOMS fqan    : [u'/lsgrid']

Proxies uploaded:
DN
↪      | Group | Until (GMT)
/DC=org/DC=terena/DC=tcs/C=NL/O=SURF B.V./CN=homer homer@surf.nl | | 2022/07/
↪31 23:54
```

Note: What does the `dirac-proxy-init` command actually do?

- It generates a *local proxy* `x509up_uXXX` in the UI `/tmp/` directory
 - It uploads this proxy to Dirac proxy server
-

And now you are ready to submit jobs to the Grid! Or copy data from and to the Grid.

2.2.3 Describe your job in a JDL file

To submit a Grid job you must describe this in a plain text file, called JDL. The JDL file will pass the details of your job to Dirac.

Warning: Make sure you have started your session and created already a *valid proxy*.

- Log in to your User Interface.
- Create a file with the following content describing the job requirements. Save it as `simple.jdl`:

```

1  [
2    Type = "Job";
3    JobName = "my_first_job";
4    Type = "Job";
5    Executable = "/bin/sh";
6    Arguments = "jobscript.sh";
7    StdOutput = "simple.out";
8    StdError = "simple.err";
9    InputSandbox = {"jobscript.sh"};
10   OutputSandbox = {"simple.out", "simple.err"};
11 ]

```

This job involves no large input or output files. It will copy the `jobscript.sh` on the Worker Node that the job will land on and execute it. The Standard output and Standard error will be directed to the files `simple.out` and `simple.err`, respectively, and retrieved when the Job Output is retrieved.

2.2.4 Submit the job to the Grid

To submit your first Grid job and get an understanding of the job lifecycle, we will perform these steps:

- *Job submission*
- *Status tracking*
- *Output retrieval*

You should have your `simple.jdl` file ready in your UI up to this point. When you submit this simple Grid job to the Dirac, a job will be created and sent to a remote Worker Node. There it will execute the script `jobscript.sh` and write its standard output and its standard error in the `simple.out` and `simple.err` respectively.

- Submit the simple job by typing in your UI terminal this command:

```
$dirac-wms-job-submit simple.jdl -f jobid
JobID = 314
```

The option `-f` allows you to specify a file (in this case `jobid`) to store the unique job identifier. Omitting the `-f` option means that the `jobID` is not saved in a file. When you do not save this id you will effectively loose the output of your job!

2.2.5 Track the job status

To check the current job status from the command line, apply the following command that queries Dirac for the status of the job.

- After submitting the job, type:

```
$dirac-wms-job-status 314
```

- Alternatively, if you have saved your jobIds into a file you can use the `-f` option and the filename as argument:

```
$dirac-wms-job-status -f jobid
```

Cancel job

- If you realise that you need to cancel a submitted job, use the following command:

```
$dirac-wms-job-delete 314
```

- Alternatively, if you have saved your jobIds into a file you can use the `-f` option and the filename as argument:

```
$dirac-wms-job-delete -f jobid
```

2.2.6 Retrieve the output

The output consists of the files included in the `OutputSandbox` statement. You can retrieve the job output once it is successfully completed, in other words the job status has changed from `Running` to `Done`. The files in the output sandbox can be downloaded for approximately one week after the job finishes.

Note: You can choose the output directory with the `-D` option. If you do not use this option then the output will be copied under the UI in the current working directory with a name based on the ID of the job.

- To get the output, type:

```
$dirac-wms-job-get-output 314
```

- Alternatively, you can use the jobid file:

```
$dirac-wms-job-get-output -f jobid
```

where you should substitute `jobid` with the file that you used to store the job ids. Please bear in mind the size of your home directory on the UI when downloading large output files. When dealing with large input and/or output files it is recommended to download the input data directly to the worker node, and upload the output data to a suitable storage space within the job itself. Please check out the `grid_storage` section for details on various clients supported on the worker nodes and best practices.

Check job output

- To check your job output, browse into the downloaded output directory. This includes the `simple.out`, `simple.err` files specified in the `OutputSandbox` statement:

```
$ls -l /home/homer/314

-rw-rw-r-- 1 homer homer  0 Jan  5 18:06 simple.err
-rw-rw-r-- 1 homer homer 20 Jan  5 18:06 simple.out

$cat /home/homer/314/simple.out
```

2.2.7 Recap & Next Steps

Congratulations! You have just executed your first job to the Grid!

Let's summarise what we've seen so far.

You interact with the Grid via the UI machine `ui.grid.surfsara.nl`. You describe each job in a JDL (Job Description Language) file where you list which program should be executed and what are the worker node requirements. From the UI, you create first a proxy of your Grid certificate and submit your job with `dirac-*` commands. The resource broker Dirac accepts your jobs, assigns them to the most appropriate CE (Computing Element), records the jobs statuses and retrieves the output.

See also:

Try now to port your own application to the Grid. Check out the [Best practices](#) section and run the example that suits your use case. The section [Advanced topics](#) will help your understanding for several Grid modules used in the [Best practices](#).

Done with the [General](#), but not sure how to proceed? We can help! Contact us at helpdesk@surfsara.nl.

ADVANCED TOPICS

3.1 Grid software

In this page we will talk about the options to run your software on the Grid worker nodes:

Contents

- *Grid software*
 - *Softdrive*
 - * *CVMFS*
 - * *Quickstart*
 - *Access*
 - *Logging in on the softdrive*
 - *Distributing an example file*
 - *Finding your files on the Grid nodes*
 - *Python on the Grid*
 - * *Softdrive anaconda*
 - *Docker*

3.1.1 Softdrive

Softdrive is the service that allows you to install software in a central place and distribute it *automagically* on the Grid. You install the software once, and it will be available on all clusters, to all users. This means that you no longer need to supply your software in your *input sandbox*, or download your software in your job.

This page is about using Softdrive on the grid infrastructure. The use of Softdrive is however not limited to grid alone, as it can equally well be applied to your own clusters, your own computer or in [cloud environments](#).

CVMFS

Softdrive is using the [CVMFS service](#) (short for CernVM File System) on the background. CVMFS is a network file system based on HTTP and optimised to deliver experiment software in a fast, scalable, and reliable way.

Quickstart

In this example, we will distribute a few small files to all nodes in the Dutch Grid. This should give you an idea of what is possible with *Softdrive*.

Softdrive works by logging in the software distribution node, and putting your files there. Next, you tell the software distribution system that you are ready installing files. These files will be made available on all nodes in the lsg and on all other nodes on the *Dutch National Grid*.

Access

Users of the National e-Infrastructure are entitled to use Softdrive without the need for a separate resource request. You can request access by sending an e-mail with your current project allocation id to helpdesk@surfsara.nl.

Logging in on the softdrive

Once access has been arranged, you can log in on the software distribution node, using your Grid UI username and password:

```
$ssh homer@softdrive.grid.sara.nl # replace homer with your username
```

In your home-directory (e.g. `/home/homer`), you will find a *README* file with detailed information about the *Softdrive* usage.

Distributing an example file

To demonstrate distributing files to all Grid nodes, create a file and a directory within your home directory:

```
# a test directory and a file
softdrive.grid.sara.nl:/home/homer$ mkdir -p test_dir
softdrive.grid.sara.nl:/home/homer$ echo "Hello world" > test_dir/hello.txt
```

To make this directory file available on all nodes on the Grid, you have to copy the `test_dir` under `/cvmfs/softdrive.nl/$USER`:

```
softdrive.grid.sara.nl:/home/homer$ cp -r test_dir /cvmfs/softdrive.nl/homer # replace_
↪homer with your username
```

- To force the update everywhere in the Grid, trigger publication by executing command:

```
publish-my-softdrive
```

Updating on all Grid nodes can take up to two hours.

Note: You need to run the command `publish-my-softdrive` each time you make a change in your `/cvmfs/softdrive.nl/$USER` directory in order to take effect on the Grid sites.

Finding your files on the Grid nodes

On Dutch Grid nodes, your Softdrive files will be available under:

```
/cvmfs/softdrive.nl/homer/ # replace homer with your username
```

Log in to your [UI account](#) and check whether your files are there:

```
ui.grid.sara.nl:/home/homer$ ls /cvmfs/softdrive.nl/homer/
drwxr-xr-x 17 cvmfs cvmfs 4096 Dec 16 12:11 test_dir
```

Note: If your software is statically compiled, then copying the executables from your home directory to `/cvmfs/softdrive.nl/$USER/` should work. Just remember to export the `/cvmfs/softdrive.nl/$USER` software paths into your Grid scripts or UI `.bashrc`. In other cases with library path dependencies, we advice you to install your software directly under `/cvmfs/softdrive.nl/$USER` or use a prefix. An example of software installation in Softdrive can be found in section [anaconda on Grid](#).

3.1.2 Python on the Grid

If you want to use a different python version to the existing on the Grid nodes or additional packages, we recommend you to install [Anaconda python](#) in your UI or [Softdrive](#) account.

Next is an example of installing the *Anaconda* python distribution in *Softdrive*.

Softdrive anaconda

- Log in to Softdrive with your account:

```
$ssh homer@softdrive.grid.sara.nl # replace homer with your username
```

- Download in your home account the [latest version of Anaconda](#) installer for linux, e.g.:

```
$wget https://repo.continuum.io/archive/Anaconda2-5.1.0-Linux-x86_64.sh
```

- Run the installer (read and approve the license terms) in Softdrive:

```
$bash Anaconda2-5.1.0-Linux-x86_64.sh
```

Note here! The installer will ask you to which location to install the software. Do not accept the default but change it to: `/cvmfs/softdrive.nl/$USER/anaconda-2-5.1.0/`:

```
Anaconda2 will now be installed into this location:
/home/homer/anaconda2
- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
```

(continues on next page)

(continued from previous page)

```
- Or specify a different location below
```

```
[/home/homer/anaconda2] >>> /cvmfs/softdrive.nl/homer/anaconda-2-5.1.0/
...
```

That was it! You can now publish the software that is installed in your `/cvmfs/softdrive.nl/homer/anaconda-2-5.1.0` directory. To do so, run this command in Softdrive:

```
$publish-my-softdrive
```

Then check after 1-2 hours from the UI if the `/cvmfs/softdrive.nl/homer/anaconda-2-5.1.0` exists.

Finally, remember to include the installation path in your scripts as:

```
$export PATH=/cvmfs/softdrive.nl/homer/anaconda-2-5.1.0/bin:$PATH # replace homer with ↪
↪your username
```

3.1.3 Docker

At the moment it is not possible to run Docker containers on the *Dutch National Grid* or lsg. We are currently investigating different possibilities and we offer *Singularity* as a container service. Please contact us at helpdesk@surfsara.nl to discuss about the available options.

3.2 Grid storage

In this page we will talk about the Grid storage facilities, the tools to interact with it and the method to handle data that is stored on tape.

Contents

- *Grid storage*
 - *About Grid storage*
 - *dCache*
 - *Grid file identifiers*
 - * *Transport URL or TURL*
 - * *Storage URL or SURL*
 - * *Default ports*
 - *Storage clients*
 - *Staging files*
 - * *gfal2 python API*
 - *Preparation*
 - *State operations*
 - *Stage operations*

- *Unpinning operations*
 - * *Monitor staging activity*
 - * *Unpin a file*
- *Checksums*
- *Transport security*
- *SRM interaction example diagram*
- *Importing large amounts of data*

3.2.1 About Grid storage

Each cluster on the Grid is equipped with a Storage Element or SE where data is stored. The Grid storage is useful for applications that handle large amount of data that can not be sent with the *job Sandbox* or stored in a *pilot job database*.

You can interact with the Grid storage from the UI or from a Worker Node, within your running job. The scripts that can access the Grid storage can be submitted from:

- *The UI*
- *The Dutch Grid*

To use the Grid storage you must:

- Have *a personal Grid certificate*¹
- Be member of *a VO* for which we have allocated storage space.

You can access the Grid storage with Grid *Storage clients*, through interfaces that speak protocols like SRM (Storage Resource Management), GRIDFTP, GSIDCAP (dCache Access Protocol with Grid authentication) or WEBDAV (Web Distributed Authoring and Versioning). With these storage clients you can:

- list directories and files
- read (download) files
- write (upload) files
- delete files or directories
- *stage* files (copy them from tape to disk for faster reading)

3.2.2 dCache

The storage element located at SURFsara is accessible from *any* Grid cluster or UI. It uses the *dCache system* for storing and retrieving huge amounts of data, distributed among a large number of server nodes. It consists of magnetic tape storage and hard disk storage and both are addressed by a common file system. See dCache-specs for details about our dCache instance.

¹ It is possible to access the dCache Grid storage without certificate, by using *webdav clients* with username/password authentication. However, authentication with username/password is less secure, and Webdav is not as fast as GRIDFTP (File Transfer Protocol with Grid authentication).

3.2.3 Grid file identifiers

You can refer to your files on the Grid with different ways depending on which of the available *Storage clients* you use to manage your files:

Transport URL or TURL

Examples:

```
# lsgrid user homer stores the file zap.tar on dCache storage
gsiftp://gridftp.grid.surfsara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar

# same, but with a Webdav TURL
https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
```

Clients for TURLs

- uberftp
- globus
- gfal
- fts
- globusonline

Storage URL or SURL

Example:

```
# lsgrid user homer stores the file zap.tar on dCache storage
srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
```

Clients for SURLs

- srm
- gfal
- fts

Default ports

Protocol	Host(s) and port(s)	Remark
SRM	srm://srm.grid.sara.nl:8443	
GridFTP	gsiftp://gridftp.grid.surfsara.nl:2811	Data channel port range: 20000-25000 See <i>webdav clients</i> for details
Webdav	https://webdav.grid.surfsara.nl:443	
	https://webdav.grid.surfsara.nl:2880	
	https://webdav.grid.surfsara.nl:2881	
	https://webdav.grid.surfsara.nl:2882	
	https://webdav.grid.surfsara.nl:2883	
	https://webdav.grid.surfsara.nl:2884	
	https://webdav-cert.grid.sara.nl:443	
GSIdCap	gsidcap://gsidcap.grid.sara.nl:22128	
xroot	xrootd.grid.sara.nl:1094	Used by CERN only
all	ipv4.grid.sara.nl	For clients that don't speak IPv6

The last one, `ipv4.grid.sara.nl`, is a single VM that supports only IPv4 and no IPv6. It can be used for small scale access through GridFTP, Webdav, Xroot or GSIdCap where IPv6 causes problems. Don't use it for batch processing.

3.2.4 Storage clients

The `InputSandbox` and `OutputSandbox` attributes in the *JDL* file are the basic way to move files to and from the User Interface (UI) and the Worker Node (WN). However, when you have large files (from about 100 MB and larger) then you should not use these sandboxes to move data around. Instead you should use the *dCache* and work with several *Storage clients*.

In this section we will show the common commands to use the various storage clients.

Note: From the many Grid storage clients, we recommend you to use the : *globus client* or *gfal client*. These tools have a clean interface, and their speed is much better on our systems compared with their `srm-*` equivalents.

Table 1: Storage clients

Client	protocols					
	SRM	GridFTP ²	GSIdCap	Webdav	3rd party	Tape control ³
<i>globus client</i>	—	yes	—	—	—	—
<i>srm client</i>	yes	⁴	⁴	⁴	yes	yes
<i>gfal client</i>	yes	yes	—	—	yes	yes
<i>webdav clients</i>	—	—	—	yes	—	—
<i>fts client</i>	yes	yes	—	yes	yes	yes
<i>globusonline client</i>	—	yes	—	—	yes	—
<i>uberftp client</i> ⁵ (not recommended)	—	yes	—	—	—	—

² The GridFTP protocol offers the best network performance.

³ Examples of tape control: staging a file from tape to disk, or get its locality (tape or disk).

⁴ SRM and LCG commands use the SRM protocol for metadata level operations and switch to another protocol like GridFTP for file transfers. This may cause protocol overhead. For example, authentication needs to be done twice: once for each protocol. For small files, that may be inefficient.

⁵ UberFTP currently has a dangerous bug that may destroy data. See https://ggus.eu/?mode=ticket_info&ticket_id=129103 for details.

uberftp client

This page includes the basic commands to use *uberftp*. For an overview of storage clients, see *Storage clients*.

Contents

- *uberftp client*
 - *Uberftp*
 - * *Creating/listing*
 - * *Transferring data*
 - *Parallel streams*
 - * *Removing data*

Uberftp

Warning: We have observed that *uberftp* deletes a file after it has unsuccessfully tried to overwrite it. If you write data with *uberftp*, please make sure you never try to overwrite existing files! *:globus-url-copy* does not have this bug.

Creating/listing

Note: To run the examples below you need to have a valid proxy, see *startgridsession*.

- Listing directories on dCache:

```
$uberftp -ls gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/
```

- Create a new directory on dCache:

```
$uberftp -mkdir gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/  
↪homer/newdir
```

Transferring data

- Copy file from dCache to local machine:

```
$uberftp gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.  
↪tar file:///home/homer/zap.tar
```

- Copy file from local machine to dCache:

```
$uberftp file:///home/homer/zap.tar gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.  
↪sara.nl/data/lsgrid/homer/zap.tar
```

Note: The asterisk “*” wildcard (match all characters) works with uberftp. Please use this option with caution, especially when deleting files.

Parallel streams

The GridFTP protocol allows for parallel streaming of data transfers. This makes a transfer more efficient and less susceptible to network errors, especially over long distances. If you have a lot of simultaneous transfers running anyway, increasing the number of streams per transfer will not make a big difference, because the network bandwidth may limit the results.

```
$uberftp -parallel 4 \
$   gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar \
$   file:zap.tar
```

Results may vary based on circumstances. We suggest a number of 4 streams as a start.

Removing data

- Remove a file from dCache:

```
$uberftp -rm gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/
↪zap.tar
```

- Remove whole (non-empty) directory with all content from dCache:

```
$uberftp -rm -r gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/
↪homer/testdir/
```

globus client

This page includes the basic commands to use globus. For an overview of storage clients, see [Storage clients](#).

Contents

- *globus client*
 - *Globus tools*
 - * *Creating/listing*
 - * *Transferring data*
 - *Parallel streams*
 - * *Removing data*
 - * *Fifo pipes*
 - *Extract directory from dCache*
 - *Extract a file*
 - *Transfer directory to dCache*

Globus tools

Note: To run the examples below you need to have a valid proxy, see `startgridsession`.

Creating/listing

- Listing directories on dCache:

```
$globus-url-copy -list gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/
↳ lsgrid/homer/
```

The `globus-*` client does not offer an option to create directories. For this purpose use a different client, e.g. *uberftp client*.

Transferring data

Note: The options `-dbg -gt 2 -vb` would show you extra logging information for your transfer.

- Copy file from dCache to local machine:

```
$globus-url-copy \
$ gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
↳ \
$ file:///`pwd`/zap.tar
```

Note: `globus-url-copy` does **NOT** encrypt the data channel when transferring data to and from dCache. Even when you supply the commandline flags `-dcpriv` or `-data-channel-private` to enforce encryption the data transfers are still not encrypted. If you need to transfer sensitive data, please contact our helpdesk. Then we can help you with a more secure alternative. This flaw has been reported to the appropriate organisations.

- Copy file from local machine to dCache:

```
$globus-url-copy \
$ file:///`pwd`/zap.tar \
$ gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
```

- Recursive upload to dCache:

```
$globus-url-copy -cd -r \
$ /home/homer/testdir/ \
$ gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/testdir/
## replace testdir with your directory
```

- Recursive download from dCache:

First create the directory locally, e.g. `testdir`.

```
$globus-url-copy -cd -r \
$    gsiftp:///gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/
↪testdir/ \
$    /home/homer/testdir/
```

- Third party transfer (between dCache sites):

First create the remote directory, e.g. `targetdir`.

```
$globus-url-copy -cd -r \
$    gsiftp:///gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/
↪sourcetdir/ \
$    gsiftp:///gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/penelope/
↪targetdir/
## note: you must include the trailing slash!
```

See also:

For dCache 3rd party transfers see also *fts client*.

Parallel streams

The `globus-url-copy` uses by default 10 parallel streams for transfers.

Removing data

The `globus-*` client does not offer an option to delete files or directories. For this purpose, use a different client, e.g. *uberftp client*.

Fifo pipes

When you want to process data from a large `tar` file (hundreds of Gigabytes) that is stored on the Grid Storage, it is possible to extract just the content without copying the complete `tar` file on the Worker Node. Similarly, you can upload a directory that will be stored in a `tar` file on the Grid storage on-the-fly. This trick saves space on the local node from keeping the double copy of the data and is possible by using the `fifo` pipes technique.

Extract directory from dCache

Extract the content of a `tar` file from the Grid storage on the worker node or UI:

```
## Create fifo for input data
$INPUT_FIFO="GRID_input_fifo.tar"
$mkfifo $INPUT_FIFO
## Extract the directory from fifo and catch PID
$tar -Bxf ${INPUT_FIFO} & TAR_PID=$!
## Download the content of the tar file, replace zap.tar with your tar file
$globus-url-copy -vb \
$    gsiftp:///gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/
↪zap.tar \
$    file:///`pwd`/${INPUT_FIFO} && wait $TAR_PID
```

Extract a file

Extract a particular from a known directory location in a tar file:

```
## Create fifo for input file
$INPUT_FIFO="GRID_input_fifo.tar"
$mkfifo $INPUT_FIFO
## Extract a particular file from fifo and catch PID
$tar -Bxf ${INPUT_FIFO} zap/filename & TAR_PID=$! # replace zap/filename with
↳ the exact location of you file in the tar
## Download the file, replace zap.tar with your tar file
$globus-url-copy -vb \
$    gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/
↳ zap.tar \
$    file:///`pwd`/${INPUT_FIFO} && wait $TAR_PID
```

Transfer directory to dCache

```
$OUTPUT_FIFO="GRID_output_fifo.tar"
$mkfifo ${OUTPUT_FIFO} # create a fifo pipe
## Push output directory to file (fifo) and catch PID
$tar -Bcf ${OUTPUT_FIFO} zap/ & TAR_PID=$! # replace zap/ with the directory
↳ to be uploaded
## Upload the final dir with fifo
$globus-url-copy -vb file:///`${PWD}`/${OUTPUT_FIFO} \
$    gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/
↳ zap.tar && wait ${TAR_PID}
## note:add stall-timeout flag in sec (e.g. -stall-timeout 7200) for large
↳ files that take too long to complete checksum on the server after transfer
```

srm client

This page explains the use of the srm client. For an overview of storage clients, see *Storage clients*.

Contents

- *srm client*
 - *SRM*
 - * *Creating/listing*
 - * *Transferring data*
 - *Recursive transfer*
 - *Parallel streams*
 - * *Removing data*
 - *Recursive delete*
 - * *Staging*

- *Staging a single file*
- *Staging a list of files*

SRM basics

See also:

Have a look at our mooc video *Storage Resource Manager* for additional examples.

SRM

The Storage Resource Manager (short SRM) has been designed to be a single interface for the management of both disk and tape storage resources. It provides options for copying files to/from the Grid storage, *Staging files* from tape, creating or removing files and so on. It uses *SURLs* as the physical filename to reference a file.

The *srm client* is one of the most popular *Storage clients*. However, srm- commands are using Java which has the tendency to allocate big amounts of memory and sometimes be slow, also because of the SRM protocol overhead. If transfer speed is important, use *uberftp client*, *globus client* or *gfal client* instead.

Note: To run the examples below you need to have a valid proxy, see startgridsession.

Creating/listing

- Listing directories on dCache:

```
$srmls srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/
```

- Create a new directory on dCache:

```
$srmmkdir srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/newdir/
```

Transferring data

Note: The -debug option would show you extra logging information for your transfers.

- Copy file from dCache to local machine:

```
## note the flag -server_mode=passive!
$srncp -server_mode=passive \
$srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar \
$file:///`pwd`/zap.tar
```

- Copy file from local machine to dCache:

```
$srncp -debug file:///`pwd`/zap.tar \
$srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
```

Recursive transfer

Recursive transfer of files is not supported with the `srm-*` client commands.

Parallel streams

Information not available yet.

Removing data

- Remove a file from dCache:

```
$srmrm srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
```

Recursive delete

Recursive deletion of files is not supported with the `srm-*` client commands. It is possible to remove a directory as long as it is empty, i.e. content files have been removed.

Staging

Staging a single file

- Check the locality status of a file:

```
$srm ls -l srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar | grep ↵  
↵ locality  
#locality:ONLINE_NEARLINE
```

- Submit a staging request for a single file. After submitting the staging command below, the prompt is waiting for the file to get online. Once the file gets online, a unique request id is returned. The pin lifetime is set in seconds, in this examples the requested pin time is a day (or 86400 sec):

```
$srm bring-online -request_lifetime=86400 srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/ ↵  
↵ data/lsgrid/homer/zap.tar  
#srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar brought online, ↵  
↵ use request id 897617461 to release
```

- Unpin a file:

After submitting the unpinning command below, the file will remain cached but purgeable until new requests will claim the available space:

```
$srm release-files srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap. ↵  
↵ tar -request_tokens=[tokenID] #replace tokenID with 897617461 retrieved above
```

- Release all pins of a file:

```
$srm release-files srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap. ↵  
↵ tar
```


Staging a list of files

Here is an example to stage a list of files. Let's say that you want to stage all the *.tgz* files in a certain dCache directory like */pnfs/grid.sara.nl/data/lsgrid/homer/*. We will use the command *srm-bring-online*. However, when you run this command the prompt will hang until the file is actually staged. So you can start first a *screen* shell to make sure your copying process continues when you accidentally loose connection to the server.

- Start screen on the UI:

```
$screen
```

- Create the file list and note that you need to use a SURL for your filepaths:

```
$FILES=`srm ls srm://srm.grid.sara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/ | egrep 'tgz'`  
↪ | awk '{print "srm://srm.grid.sara.nl:8443"$2}'`
```

- Check if the list looks OK. All filenames should be split with spaces in one line:

```
$echo $FILES  
#srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/file1.tgz srm://srm.  
↪ grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/file2.tgz srm://srm.grid.sara.  
↪ nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/file3.tgz srm://srm.grid.sara.nl:8443/pnfs/  
↪ grid.sara.nl/data/lsgrid/homer/file4.tgz
```

- Check the status of the files to see how many are online (Disk) and how many are nearline (Tape):

```
$srm ls -l $FILES | grep locality | awk '/ONLINE/{i++};/:NEARLINE/{j++}; END{print "Disk:  
↪ ", i, ", Tape:", j}'  
#Disk: , Tape: 4
```

- Submit the stage command to request staging the bulk of files. You can store the output in a file *stage.log* to save the request IDs. The pin lifetime here is set to 1 week, but note that this counts from the moment you submit the request independent to the actual time that the files are on disk:

```
$srm-bring-online $FILES -request_lifetime=604800 > stage.log  
# prompt will hang until operation is complete for all the files
```

- Once processing of the requested files is done, you can release the bulk of files so that the pin is removed and the staging read-pool is free for other data:

```
$srm-release-files $FILES
```

gfal client

This page includes the basic commands to use *gfal*. For an overview of storage clients, see *Storage clients*.

Contents

- *gfal client*
 - *gFAL*
 - * *Creating/listing*

- * *Transferring data*
 - *Recursive transfer*
 - *FIFO transfers*
- * *Removing data*
- * *Checksums*
- * *Locality*

gFAL

Note: To run the examples below you need to have a valid proxy, see `startgridsession`.

Mandatory environment settings:

```
$export LCG_GFAL_INFOSYS=bdii.grid.sara.nl:2170
```

Note: The examples below will work both with *TURLs* and *SURLs*.

Creating/listing

- Listing directories on dCache:

```
$gfal-ls -l gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/
```

- Create a new directory on dCache:

```
$gfal-mkdir gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/  
↪newdir/
```

Transferring data

- Copy file from dCache to local machine:

```
$gfal-copy gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/  
↪zap.tar file:///`pwd`/zap.tar
```

- Copy file from local machine to dCache:

```
$gfal-copy file:///`pwd`/zap.tar gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.  
↪nl/data/lsgrid/homer/zap.tar
```

Recursive transfer

Recursive transfer of files is not supported with the `gfal-copy` command.

FIFO transfers

Using a fifo pipe allows you to tar/untar a file on the fly and use half of the space on the WN, e.g. tarring a directory and uploading the tarball to dCache would require 2x(directory size) space on the WN without a fifo.

- Download a tar ball from dCache using fifo

```
INPUT_FIFO="GRID_input_fifo.tar"
mkfifo ${INPUT_FIFO}
tar -Bxf ${INPUT_FIFO} & TAR_PID=$!
gfal-copy gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar ↵
↪ file:///${PWD}/${INPUT_FIFO} && wait ${TAR_PID}
```

- Upload a tar ball to dCache using fifo

```
OUTPUT_FIFO="GRID_output_fifo.tar"
tar -Bcf ${OUTPUT_FIFO} ${PWD}/mydir/ & TAR_PID=$!
gfal-copy file:///${PWD}/${OUTPUT_FIFO} gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.
↪ sara.nl/data/lsgrid/homer/zap.tar && wait ${TAR_PID}
```

Removing data

- Remove a file from dCache:

```
$gfal-rm gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
```

- Remove whole (non-empty) directory with all content from dCache:

```
$gfal-rm -r gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/
↪ testdir/
```

Checksums

- Get checksum for a file on dCache:

```
$gfal-sum gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar ↵
↪ ADLER32
```

Locality

- Get locality (ONLINE, NEARLINE) for a file on dCache:

```
$gfal-xattr gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsguid/homer/zap.  
↪tar user.status
```

webdav clients

This page describes how to use the webdav protocol. For an overview of storage clients, see [Storage clients](#).

Contents

- *webdav clients*
 - *About WebDAV*
 - *Available WebDAV doors*
 - * *Choosing a WebDAV door*
 - *Authentication*
 - *Transport security*
 - *Performance*
 - *Firewall issues*
 - *IPv4 / IPv6*
 - *Clients*
 - *Web browsers*
 - *Curl & wget*
 - * *Creating directories*
 - * *Transferring data*
 - *Uploading*
 - *Downloading*
 - *Downloading with proxy authentication*
 - *Partial downloads*
 - * *Renaming*
 - *With proxy authentication*
 - *With username/password authentication*
 - * *Removing data*
 - * *Querying file properties*
 - *Locality*
 - *Adler32 checksums*
 - *MD5 checksums*

- *Rclone*
- *Sharing data with Macaroons*
- *Graphical access with Cyberduck*
 - * *Cyberduck with a user certificate*
 - * *Mounting WebDAV with Mountain duck*

About WebDAV

The WebDAV protocol has the following advantages:

- It supports not only x509 (user certificate or proxy) authentication, but also username & password authentication and *macaroon* authentication
- It uses the common port 443. Some overly strict firewalls may block outgoing traffic, but port 443 is so common that it is seldom blocked. However, using WebDAV to bypass firewalls should be seen as a temporary solution; it would be better to open up your institute's firewall to allow access to the dCache subnet.
- It can transfer data over a secure channel; other protocols like GridFTP do authentication over a secure channel but transfer data unencrypted.

It also has disadvantages:

- It is not a high performance transfer protocol. If this is important, use GridFTP instead.
- Support by WebDAV clients varies widely. Some operations (like renaming a file) may not work in certain clients and circumstances. Modifying/overwriting a file, although the WebDAV protocol supports it, doesn't work on dCache; you'll need to delete the old file and upload the new file instead.

Available WebDAV doors

dCache has the following WebDAV doors:

URL including port	Authentication method	Remarks
https://webdav.grid.surfsara.nl:443	User/password, macaroon	Redirects on read
https://webdav.grid.surfsara.nl:2880	User/password, macaroon, X509	No redirects
https://webdav.grid.surfsara.nl:2881	User/password, macaroon	No redirects; maximum transport security
https://webdav.grid.surfsara.nl:2882	X509, macaroon	Redirects on read and write
https://webdav.grid.surfsara.nl:2883	X509, macaroon	No redirects
https://webdav.grid.surfsara.nl:2884	X509, macaroon	No redirects; maximum transport security
https://webdav-cert.grid.sara.nl:443	X509, macaroon	No redirects; maximum transport security
https://ipv4.grid.surfsara.nl:*	Same as webdav.grid.surfsara.nl, but IPv4 only	

Choosing a WebDAV door

Authentication

The most important consideration is whether you want to authenticate with username/password or with x509 (certificate/proxy). See the above table. All WebDAV doors support *macaroons* (bearer tokens).

Transport security

Another important consideration is whether a WebDAV door should redirect or not; this affects transport security and throughput.

Advantages of redirects:

- It's a form of load balancing, which improves the speed.
- Redirecting WebDAV doors do the authentication over HTTPS, but they redirect your client to an HTTP port. So the data transfer is unencrypted. This too improves speed.

Disadvantages of redirects:

- File transfers are sent over HTTP, so they are not encrypted. A “man in the middle” might be able to read the data, or even modify it in transit. If privacy is a concern, choose a door that does not redirect.
- Some WebDAV clients don't handle redirects very well.

If transport security is a requirement, we suggest to use the WebDAV doors on port 2881 or 2884 for the best transport security. They use only TLSv1.2 or better, and they are configured with encryption ciphers that offer Perfect Forward Secrecy. They have some extra HTTP security headers that may make abuse more difficult.

There might be some clients that have difficulties connecting to these high security doors. If data security is important to you, we suggest you find clients that do support these improved security settings. In the future, these settings will be applied to ports 2880 and 2883 as well.

Performance

Another consideration is whether you're using the door for parallel access.

`webdav.grid.surfsara.nl` is a DNS round robin that will direct you to a (more or less) random host in a pool of WebDAV servers. So it is very well suited for parallel access. The other host names are not powered by a group of nodes, so they are less suited to parallel processing.

Firewall issues

Use `webdav-cert.grid.sara.nl` when you want to authenticate with a user certificate or proxy, and your institute's firewall blocks outgoing connections to port 2882 to 2884. It's a single virtual machine; don't use it for parallel processing. It is configured with the best security settings, like the ones on port 2881 and 2884.

IPv4 / IPv6

All WebDAV doors are dual stack: they support both IPv4 and IPv6, with IPv6 as the preferred protocol. Use `ipv4.grid.surfsara.nl` for storage clients that have problems with IPv6. It's a single virtual machine; don't use it for parallel processing.

Clients

We've tested these WebDAV clients successfully with dCache:

- web browsers (read only)
- curl
- wget (read only)
- rclone (username/password; no x509 authentication)
- cyberduck (GUI)

We'll describe how to use them below.

Web browsers

The easiest way to access dCache is with a normal web browser. You can point a browser like Firefox to <https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/> or any of the other WebDAV doors listed in the table above. When the browser asks for a username and password, you can provide your Grid UI (or CUA (SURFsara's Central User Administration)) username and password. An icon in front of each file indicates the locality of the file (tape or disk). When you click on a listed file, it will be downloaded, if you're authorized to do so.

You can't upload to dCache WebDAV with a normal browser. There is a new interface, [dCache View](#), that allows you to upload files from your browser. This is still in development, so test it first before using it for production data.

Curl & wget

Note: To run the examples below you need to have a UI (or CUA) account that is configured within dCache and authorized to the data you want to access. Contact us if you need assistance with that.

Creating directories

To create a directory with curl:

```
$curl --capath /etc/grid-security/certificates/ --fail --user homer \  
--request MKCOL https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsguid/homer/  
↪directory
```

If on your system there are no Grid CA certificates available in `/etc/grid-security/certificates/`, please read [host_certificates](#).

Transferring data

Uploading

To copy a file from your local machine to dCache:

```
$ curl --capath /etc/grid-security/certificates/ --fail --location --user homer \
    --upload-file zap.tar \
    https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/
$# replace homer with your username, lsgrid with your VO and zap.tar with your local file
```

The command will ask for the password of ‘homer’ on the command line. If you don’t want to type the password each time, specify `--netrc` and store the password in the `.netrc` file in your home dir. Make sure it is not readable by others (`chmod 600 .netrc`). See `man curl` for more details. An example `.netrc` file is:

```
$ cat .netrc
machine webdav.grid.surfsara.nl
login <your_username>
password <your_pwd>
```

Note: It is possible to specify the password on the command line like this: `--user homer:password`. However, for security reasons this should be avoided on shared systems (like the UI) because it allows other local users to read the password with the `ps` command.

Downloading

To copy a file from dCache to your local machine:

```
$ curl --capath /etc/grid-security/certificates/ --fail --location --user homer \
    https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar \
    --output zap.tar
```

Or with `wget`:

```
$ wget --user=homer --ask-password --ca-directory=/etc/grid-security/certificates \
    https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
```

The `--ca-directory` is probably not necessary anymore. If it is, but you don’t have an `/etc/grid-security/certificates` directory, you could specify `--no-check-certificate`, but we don’t recommend this.

Downloading with proxy authentication

To download a file while using a proxy to authenticate, you first have to create your proxy, see `startgridsession`.

Then use a command like this:

```
$ curl --location --capath /etc/grid-security/certificates/ \
    --cert $X509_USER_PROXY --cacert $X509_USER_PROXY \
    https://webdav.grid.surfsara.nl:2882/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar
```


Note: It is possible that your proxy DN (Distinguished Name) is mapped to another user account than your own CUA user account. If you have permission issues with either username or proxy and not the other, contact us to check the user mapping.

Note: `wget` does not support certificate/proxy authentication.

Partial downloads

With Curl you can download only part of a file by specifying `--range`. Example:

```
$curl --fail --location --capath /etc/grid-security/certificates/ \
--user homer \
https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/myfile \
--range 0-4 \
--output first-5-bytes
```

Renaming

With proxy authentication

```
$curl --capath /etc/grid-security/certificates/ --fail --location \
--cert $X509_USER_PROXY --cacert $X509_USER_PROXY \
--request MOVE \
https://webdav.grid.surfsara.nl:2882/pnfs/grid.sara.nl/data/lsgrid/homer/oldfile \
--header "Destination:https://webdav.grid.surfsara.nl:2882/pnfs/grid.sara.nl/data/
↪lsgrid/homer/newfile"
```

File properties and locality are not changed. A file that is stored on tape (nearline) will stay on tape, even if it is moved to a directory for disk-only files.

With username/password authentication

```
$curl --capath /etc/grid-security/certificates/ --fail --location-trusted \
--user homer \
--request MOVE \
https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/data/lsgrid/homer/oldfile \
--header "Destination:https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/data/
↪lsgrid/homer/newfile"
```

Please note the differences with the previous example:

- `--location-trusted` will send the username and password also to the destination server.
- Port 2880 is used for username/password authentication.

Removing data

Deleting a file from dCache:

```
$curl --capath /etc/grid-security/certificates/ --user homer --location \
      --request DELETE https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/
↪homer/zap.tar
```

Querying file properties

With curl and a dCache WebDAV door, it's possible to request file properties. This works both with username/password and proxy authentication, provided you use the correct port (443 or 2880 for username/password, 2882 or 2883 for proxy authentication).

Locality

This example shows how to query the file locality: whether a file is online or nearline (on tape). This example uses username/password authentication:

```
$echo -e '<?xml version="1.0"?>\n
      <a:propfind xmlns:a="DAV:">
      <a:prop><srm:FileLocality xmlns:srm="http://srm.lbl.gov/StorageResourceManager"/
↪></a:prop>
      </a:propfind>' \
| curl --silent --fail --capath /etc/grid-security/certificates/ \
      --user homer --request PROPFIND \
      https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar \
      --header "Content-Type: text/xml" --upload - \
| xmllint -format -
```

See *Staging files* for more information about file locality.

Adler32 checksums

This example shows how to get the Adler32 checksum of a stored file. dCache uses Adler32 checksums by default, but this can be configured per project.

The returned checksum comes from the dCache database, so it is a very efficient way to check your files. dCache does checksum checks on most operations, so you can safely assume the checksum matches the stored file.

```
$curl --head --header 'Want-Digest: ADLER32' --silent --fail --capath /etc/grid-security/
↪certificates/ \
      --user homer \
      https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/myfile \
| grep 'adler32='
```

Here an example output:

```
Digest: adler32=46fd067a
```

Here is an alternative way to query an Adler32 checksum:

```
$echo -e '<?xml version="1.0"?>\n
    <a:propfind xmlns:a="DAV:">
    <a:prop><srm:Checksums xmlns:srm="http://www.dcache.org/2013/webdav"/></a:prop>
    </a:propfind>' \
| curl --silent --fail --capath /etc/grid-security/certificates/ \
    --user homer --request PROPFIND \
    https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/myfile \
    --header "Content-Type: text/xml" --upload - \
| xmllint -format - \
| egrep -o '<ns1:Checksums>.*</ns1:Checksums>'
```

Here is an example of the expected output:

```
$<ns1:Checksums>adler32=46fd067a</ns1:Checksums>
```

There's a script that uses the above technique to retrieve checksums: <https://github.com/onnozweers/dcache-scripts/blob/master/get-file-checksum>

MD5 checksums

dCache is configured to use Adler32 checksums by default. However, in some cases, dCache may have a file's MD5 checksum in its database.

You can use WebDAV to retrieve the MD5 checksum of a file, when it is in dCache's database. It's a bit more complicated than Adler32 because MD5 checksums are presented in base64 encoding, as prescribed by RFC 3230.

```
$curl --head --header 'Want-Digest: MD5' --silent --fail --capath /etc/grid-security/
certificates/ \
    --user homer \
    https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/lsgrid/homer/myfile \
| grep -o 'md5=.*' \
| sed -e 's/md5=/' -e 's/[\r\n]*$//' \
| base64 --decode \
| xxd -p
```

The output should look similar to this:

```
0f43fa5a262c476393018f7329080fa7
```

An alternative way to query an MD5 checksum:

```
$echo -e '<?xml version="1.0"?>\n
    <a:propfind xmlns:a="DAV:">
    <a:prop><srm:Checksums xmlns:srm="http://www.dcache.org/2013/webdav"/></a:prop>
    </a:propfind>' \
| curl --silent --fail --capath /etc/grid-security/certificates/ \
    --user homer --request PROPFIND \
    https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/lsgrid/homer/myfile \
    --header "Content-Type: text/xml" --upload - \
| xmllint -format - \
| egrep -o '<ns1:Checksums>md5=.*</ns1:Checksums>' \
| sed -e 's#<ns1:Checksums>[^=]*=([<]*\<)/ns1:Checksums>#\1#' \
```

(continues on next page)

(continued from previous page)

```
| base64 --decode \  
| xxd -p
```

Queries can be combined to reduce transaction overhead:

```
$echo -e '<?xml version="1.0"?>\n  
  <a:propfind xmlns:a="DAV:">  
    <a:prop><srm:RetentionPolicy xmlns:srm="http://srm.lbl.gov/  
→StorageResourceManager"/></a:prop>  
    <a:prop><srm:AccessLatency xmlns:srm="http://srm.lbl.gov/StorageResourceManager  
→"/></a:prop>  
    <a:prop><srm:FileLocality xmlns:srm="http://srm.lbl.gov/StorageResourceManager"/  
→></a:prop>  
    <a:prop><srm:Checksums xmlns:srm="http://www.dcache.org/2013/webdav"/></a:prop>  
  </a:propfind>' \  
| curl ...
```

There's a script that uses the above technique to retrieve checksums: <https://github.com/onnozweers/dcache-scripts/blob/master/get-file-checksum>

Rclone

Rclone is a command line tool that you can download from <https://rclone.org/downloads/>. It works on many platforms and it can talk to many storage systems besides WebDAV.

Advantages of Rclone are:

- It can sync directories, like rsync does
- For directories, it uses parallel transfers, 4 by default, to get a better performance

For authentication, Rclone can use username/password (from the CUA) or token based (macaroon, see below) authentication, but not X509 certificate/proxy authentication. If you have a Grid certificate and want to use Rclone, you can use `get-macaroon` with your Grid certificate or proxy to create an Rclone config file with a macaroon for authentication; see the chapter below.

Note: New versions of Rclone will use multiple streams for files that are larger than 200 MB, but dCache may not support that. Please use Rclone with `--multi-thread-streams 1`. Rclone will then only use one stream per file, but it will still do 4 files in parallel when copying directories.

Note: The default idle timeout in Rclone is 5 minutes. This may be too short for the checksum calculation phase when uploading large files (>10GB). You can increase it with `--timeout=240m`.

Note: When uploading files from multiple Rclone clients in parallel, please add `--no-traverse` as argument. Rclone by default does a directory listing before upload. When this is done in parallel for large directories, it can be bad for performance.

The first time you use rclone, you need to make a profile with `rclone config`.

As the remote URL, you can use for example `https://webdav.grid.surfsara.nl:443/pnfs/grid.sara.nl/data/lsgrid/homer` (for performance) or `https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/data/lsgrid/homer` (with encrypted transport).

An example of a profile:

```
[dcache]
url = https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/data/lsgrid/homer
vendor = other
user = homer
pass = *** ENCRYPTED ***
```

An example of using rclone to copy a directory:

```
$rclone --multi-thread-streams 1 --timeout=240m --no-traverse copy mydir dcache:rclone-
↪test
```

More information on how to use rclone with WebDAV is here: <https://rclone.org/webdav/>. There are also graphical user interfaces to rclone; one is [RcloneBrowser](#).

Sharing data with Macaroons

Macaroons are bearer tokens that authorize someone to access certain directories or files. With this technique, you can share (some of) your data with anyone else. The other person does not need to have a user account or a certificate; only a WebDAV client that supports bearer tokens. Clients that support this are Curl, Rclone and (read only) ordinary browsers such as Firefox. Cyberduck does not support it (yet).

A Macaroon may contain caveats that limit access. Such caveats can be based on the data path, the activities that may be performed with the data (list, download, upload, etc.), the IP address of the client, or a maximum validity period.

Warning: Always add sufficient caveats to your Macaroons to avoid theft and abuse!

For your convenience, we've created a script called `get-macaroon` that makes it easy to obtain a Macaroon. It's installed on the UI. Example:

```
12:12 ui.grid.surfsara.nl:/home/homer
homer$ get-macaroon --url https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/data/
↪lsgrid/homer/Shared/ --chroot --user homer --duration PT1H --permissions DOWNLOAD,LIST
Enter host password for user 'homer':
https://webdav.grid.surfsara.nl:2880/?
↪authz=MDAxY2xvY2F0aW9uIE9wdGlvbmFsLmVtCHR5CjAwMThpZGVudGhmaWVyaWQ6MzEwMjk7MzE
↪oe2VxwSpym6rPP7fNKprXTQEh2qlXwaLKACg
```

The printed link can be pasted into a browser's address bar, or provided as an argument to curl for download.

When uploading (or downloading) with curl, the token can be provided in a custom header (replace <token> with the Macaroon):

```
$curl --header 'Authorization: BEARER <token>' --upload-file myfile https://webdav.grid.
↪surfsara.nl:2880/
```

The script can also create an Rclone config file:

```
$get-macaroon --url https://webdav.grid.surfsara.nl:2880/pnfs/grid.sara.nl/data/lsgrid/
↳homer/Shared/ --chroot --user homer --duration PT1H --permissions DOWNLOAD,LIST --
↳output rclone homers-share
Enter host password for user 'homer':
Creating rclone config file homers-share.conf:
....
Send this file to the persons you want to share data with.
They need rclone v1.42-012-gfa051ff9 or newer to access the data.
Example command:
rclone --config=homers-share.conf ls homers-share:
```

You can get a Macaroon with X509 authentication too. Please note, that port 2883 is used for this. The lifetime of your proxy does *not* limit the lifetime of the macaroon.

```
$voms-proxy-init -voms lsgrid:/lsgrid
Enter GRID pass phrase for this identity:
....
Your proxy is valid until Fri Jul 06 01:37:31 CEST 2018

$get-macaroon --url https://webdav.grid.surfsara.nl:2883/pnfs/grid.sara.nl/data/lsgrid/
↳homer/Shared --proxy --chroot --duration PT1H
https://webdav.grid.surfsara.nl:2883/?
↳authz=MDAxY2xvY2F0aW9uIE9wdGlvbmFsLmVtcHR5CjAwMThpZGVudGhmaWVyIGNOMDBnRHRSCjAwMzZjaWQgaWQ6MzY0OTQ7MzE
↳zh2OGkwo
```

For more information, see the dCache User Guide: <https://dcache.org/manuals/UserGuide-7.2/webdav.shtml#requesting-macaroon>

Graphical access with Cyberduck

To work with WebDAV on Windows or Mac OS X, you can install **Cyberduck** from here: <https://cyberduck.io/>. Please note that the App store package costs money; the download from the website is free, but will ask for a donation.

- Download the .zip file, open it, and drag the .app file into your Applications folder to install it.
- Open a WebDAV (HTTP/SSL) connection and connect to the server with your UI account username and password:

```
https://webdav.grid.surfsara.nl/pnfs/grid.sara.nl/data/lsgrid/ # replace lsgrid
↳with your VO
```

webdav.grid.sara.nl – WebDAV (HTTPS)

WebDAV (HTTP/SSL)

Nickname: webdav.grid.sara.nl – WebDAV (HTTPS)

URL: <https://homer@webdav.grid.sara.nl/data/users/homer>

Server: webdav.grid.sara.nl Port: 443

Username: homer

☐ Anonymous Login

▼ More Options

Path: /pnfs/grid.sara.nl/data/users/homer

Connect Mode: Default

Encoding: Default

☐ Use Public Key Authentication
No private key selected

Download Folder: Downloads

Transfer Files: Default

Web URL: <http://webdav.grid.sara.nl>

Notes:

Timezone: UTC

Cyberduck with a user certificate

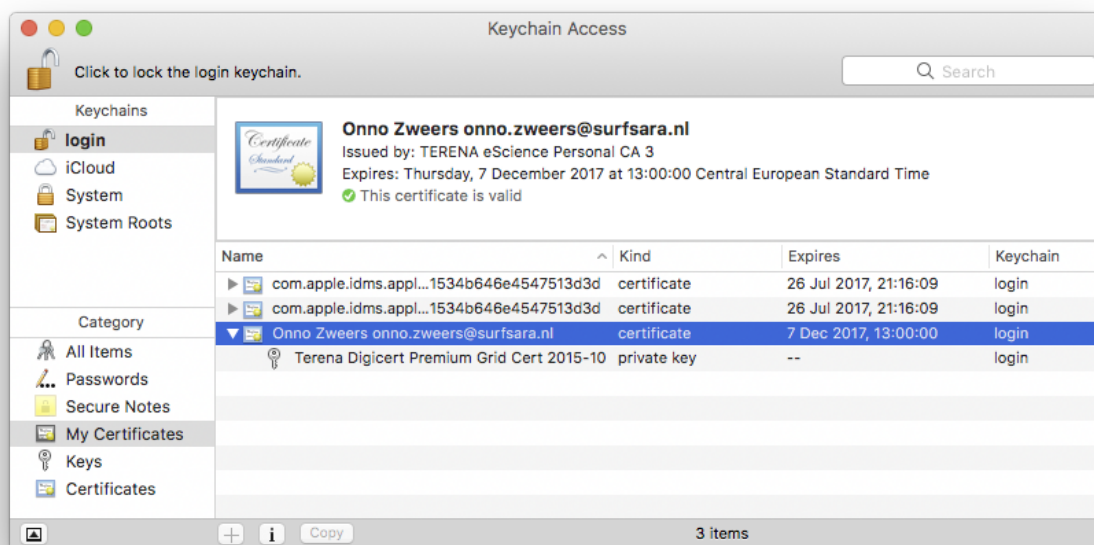
Normally, one would authenticate to dCache using a user certificate or proxy. dCache determines your identity based either on your user certificate or proxy DN, or on your VOMS credentials. However, if you authenticate with your CUA username & password, that identity might not be the same and you may not have access to your own data.

To work around this, it may be useful to have Cyberduck authenticate you using your user certificate.

Note: Most users are authenticated based on the VOMS credentials of their proxy. Since you will not use a *VOMS* proxy but a certificate, this identity mapping won't work and you may not have access. Instead, we may need to map your *DN* onto the desired identity instead of your VOMS credentials. If you want to use Cyberduck with certificate authentication, contact us so that we can map your DN to the desired identity.

Here is how to configure Cyberduck for certificate authentication on OS X:

First, import your user certificate in p12 format into the Keychain Access. It should look something like this:



Second, go to Cyberduck and create a bookmark with these settings:

webdav.grid.sara.nl cert

WebDAV (HTTP/SSL)

Nickname: webdav.grid.sara.nl cert

URL: <https://anonymous@webdav.grid.sara.nl:2882/pnfs/grid.sara.nl/data> ⚠

Server: webdav.grid.sara.nl Port: 2882

Username: anonymous

☒ Anonymous Login

▼ More Options

Path: /pnfs/grid.sara.nl/data/

Connect Mode: Default

Encoding: Default

☐ Use Public Key Authentication
No private key selected

Download Folder: Downloads

Transfer Files: Default

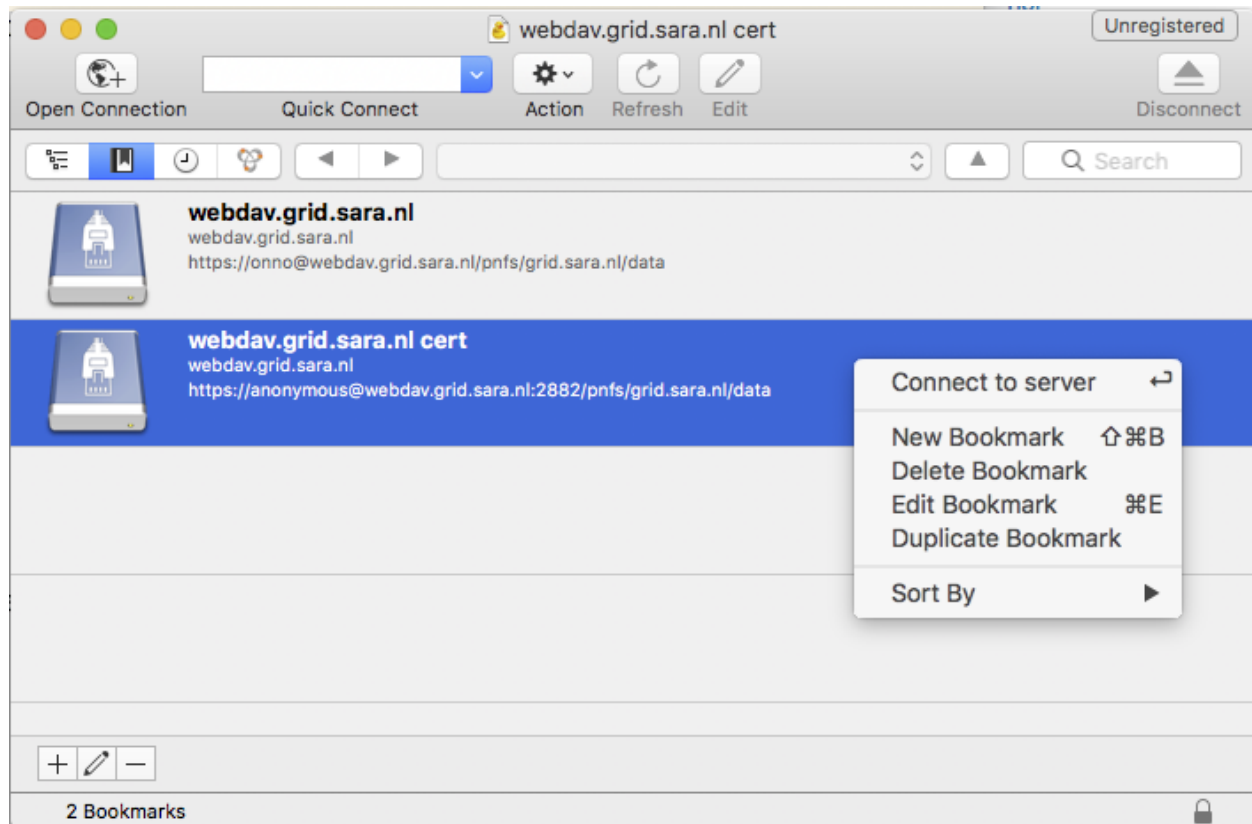
Web URL: <http://webdav.grid.sara.nl>

Notes:

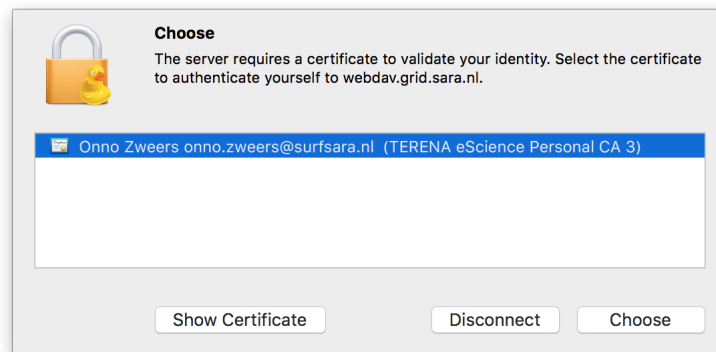
Timezone: UTC

If your institute blocks outgoing traffic to port 2882 and 2883, you can use server `webdav-cert.grid.sara.nl` and port 443, as described at the top of this page.

Right-click the bookmark and choose “Connect to server”.



Choose your certificate.



Mounting WebDAV with Mountain duck

Mountain Duck is a WebDAV client that can mount WebDAV in OS-X and Windows. We've had some mixed results using it with dCache, so you should test it before trusting it with your data. You can find it at <https://mountainduck.io/>.

fts client

This page includes the basic commands to use the FTS (File Transfer Service). For an overview of storage clients, see *Storage clients*.

Contents

- *fts client*
 - *FTS*
 - * *Introduction*
 - * *SURFsara FTS instance*
 - * *Authentication*
 - * *FTS file transfers*
 - * *Monitor Status*
 - * *Failed transfers*
 - * *Using the API with curl*

FTS

Introduction

FTS Cern is a file transfer service for reliable file transfers across sites or else, *third party* transfers. FTS queues, schedules and performs transfers, retrying it if necessary. You cannot use it to transfer files from your local machine to dCache and vice-versa. FTS service has been running for years. It was developed by Cern as the service responsible for distributing the majority of LHC data across the WLCG infrastructure, up to petabytes per month.

From the user perspective, it allows data movement, retrying if necessary, monitoring and displaying usage statistics, see [FTS wiki](#). From the operations view, it optimises resource usage.

The wisdom of FTS is the *Optimiser* which decides whether to increase or decrease the amount of transfers attempted at a time. It evaluates any symptoms of saturation and decreases the number of parallel transfers when there is an increase of recoverable errors, or the throughput worsens.

FTS has a REST API and supports several transfer protocols, such as: GridFTP, SRM, Webdav/HTTPS, xroot. It also supports several clients, such as: standard clients (e.g. curl), Python bindings, FTS CLI.

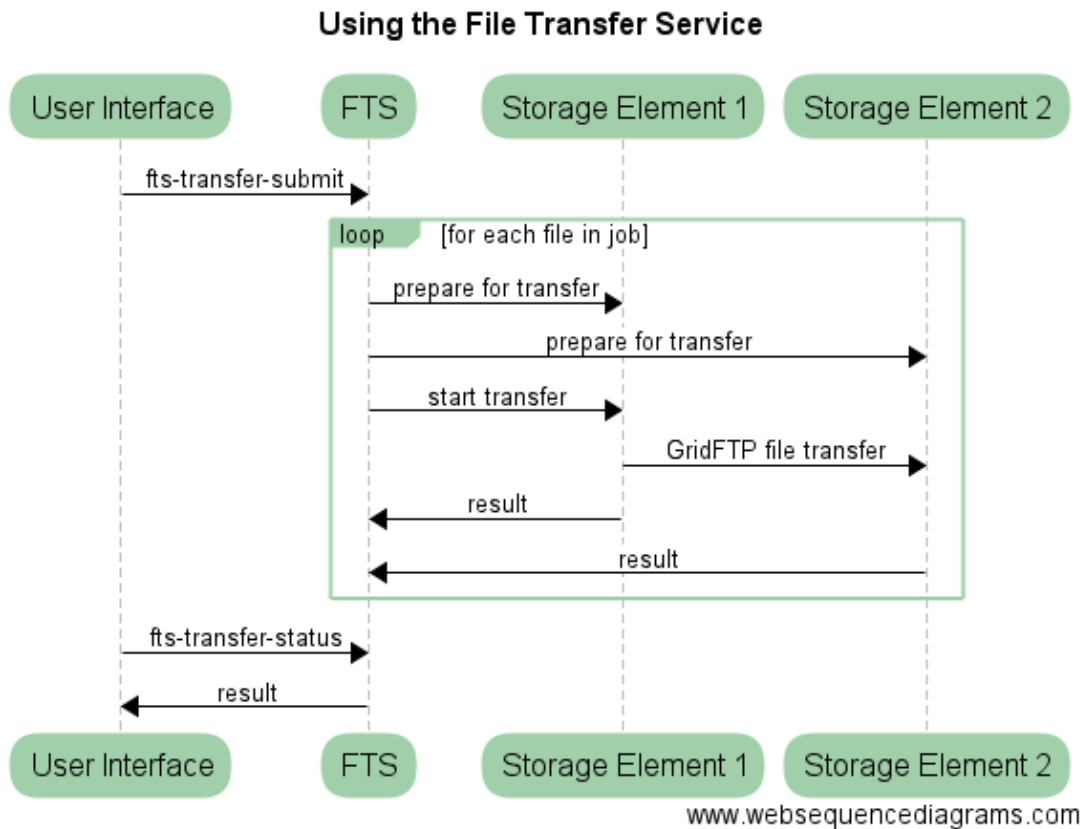
SURFsara FTS instance

Our FTS instance can be found at: [SURFsara FTS UI](#). The site can be accessed when you have a certificate installed in your browser. The FTS UI provides a sophisticated and easy to use web interface for transfer management and monitoring.

The SURFsara FTS Rest API can be used from command-line clients or python bindings. Currently we do not support the WebFTS browser view. The Cern website offers several examples to interact with the API with python, see [FTS Easy Bindings](#).

The FTS command-line client is currently installed on the UI `ui.grid.surfsara.nl` and the API can be reached in port 8446.

The following graph depicts the FTS transfer lifecycle:



Authentication

To use the FTS (File Transfer Service) you need to create a local proxy. The `fts submit` command automatically delegates the proxy to the FTS server. The default lifetime is 12 hours. This means that you need to submit an `fts` command at least once every 12h to renew the delegation, e.g. by checking the status of the transfers with an `fts-transfer-status` command. When the remaining lifetime of the stored proxy passes under 4 hours, `fts-transfer-submit` will automatically delegate a new one as long as there is a valid *local proxy*.

In our FTS instance that can be found at: [SURFsara FTS UI](#) we limit the access to only some VOs, upon request. You will need a proxy with a `voms` extension before you can use the rest interface.

Note: To run the examples in this page you need to have a valid local proxy. The `voms-proxy-init` tool can be used

to generate a proxy with VOMS (Virtual Organization Membership Service) attributes from the personal certificate. Alternatively, you can use the `startGridSession` tool available on the SURFsara UIs. See also `startgridsession-explained`.

First, create a proxy with your VO attributes on the UI:

```
$startGridSession lsgrid #replace lsgrid with your own VO
```

FTS file transfers

Submit a transfer

The `fts-transfer-submit` command submits transfer-jobs by specifying the source and destination file location. The file location can be a `SURL` (Storage URL), `TURL` (Transport URL) or `HTTPS` link. For efficient usage of the service, it is preferred that the source and destination endpoints are `GridFTP` or `SRM` servers.

The output of the submit command is a *unique ID* that can be used for tracing the transfer status.

When the FTS transfer starts, it creates the target file with 0 bytes. If the transfer fails the target file is removed independent to the amount of bytes that had already been transferred.

Note: We have noticed that using `SURLs` instead of `TURLs` slightly increases the performance (due to the `SRM` load balancer).

Basic options

Here are some basic options to use when initiating FTS transfers. The proposed values for retries, parallel streams and timeout settings depend on the amount of files and volume of data to be transferred. If you need help to estimate these values, please contact us at helpdesk@surfsara.nl

- `-v`: enable verbose information
- `-s https://fts.grid.surfsara.nl:8446`: specify the fts server.
- `-K`: enable checksum. By default, `adler32` is supported on the SURFsara servers.
- `--retry 2 --retry-delay 300`: in case of errors (timeouts, overwriting, etc) the file transfer will be retried after 5 minutes
- `--nostreams 4`: the longer the distance between the transfer endpoints, the more streams you need to achieve transfers less vulnerable to congestion
- `--timeout 14400`: this option helps in case of large file transfers to make sure that the connection will not timeout before the transfer is complete. If you omit this option, the default timeout is 4000 sec

File transfer - TURL to TURL

```
$fts-transfer-submit -s https://fts.grid.surfsara.nl:8446 \  
$   gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar \  
$   gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/penelope/zap.tar  
#641d3436-8af1-11eb-ad12-fa163e7fa8c6
```

File transfer - SURL to SURL

```
$fts-transfer-submit -s https://fts.grid.surfsara.nl:8446 \  
$   srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/zap.tar \  
$   srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/penelope/zap.tar
```

File transfer - SURL to TURL

```
$fts-transfer-submit -s https://fts.grid.surfsara.nl:8446 \  
$   srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/zap.tar \  
$   gsiftp://gridftp.grid.sara.nl:2811/pnfs/grid.sara.nl/data/lsgrid/penelope/zap.tar
```

Note: Combinations between TURLS, SURLS, HTTPS and SRMv2 are possible.

Bulk transfers

If you have multiple files to transfer, you can submit the transfers in one bulk operation. Example:

```
$fts-transfer-submit -s https://fts.grid.surfsara.nl:8446 -f transfer-list.txt
```

The list of transfers should have this format:

```
file1-source-SURL-or-TURL file1-destination-SURL-or-TURL  
file2-source-SURL-or-TURL file2-destination-SURL-or-TURL  
...
```

An example:

```
srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/file1 srm://srm.grid.  
↪sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/penelope/file1  
srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/homer/file2 srm://srm.grid.  
↪sara.nl:8443/pnfs/grid.sara.nl/data/lsgrid/penelope/zap.tar/file2
```

More information and examples of bulk transfers and FTS in general can be found at [CERN FTS documentation](#) and [Cern FTS Git repo](#).

Monitor Status

Command line

The `fts-transfer-submit` command will return instantly an ID for the specific job. This ID can be used to trace the status of the transfer:

```
$fts-transfer-status -s https://fts.grid.surfsara.nl:8446 9e665677-76e5-4734-b729-
↪b69e161da99a
## replace the string '9e665677-76e5-4734-b729-b69e161da99a' with your transfer job ID
```

For bulk transfers, monitor the status overview of all submitted files with:

```
$fts-transfer-status -s https://fts.grid.surfsara.nl:8446 -list 9e665677-76e5-4734-b729-
↪b69e161da99a | grep State: | sort | uniq --count
```

Web interface

Alternative to the `fts-transfer-status` command, you can use the graphical interface in [SURFsara FTS UI](#) to monitor the status and trace the logging information.

Generated at 10:48:06
Overview
Jobs
Optimizer
Error reasons
Statistics
Configuration
Job ID

Transfer '9a718fee-8af3-11eb-ad12-fa163e7fa8c6' FINISHED

VO: lofar/user/sksp

Delegation ID: 3d3ce1e83def1afb
 Submitted time: 2021-03-22T09:47:23Z
 Job finished: 2021-03-22T09:47:27Z
 Priority: 3
 Bring online: -1

Received by fts.ms4.surfsara.nl
 Overwrite flag:
 Job type: N
 Cancel flag:
 Pin lifetime: -1

Metadata:

null

Total size	Done	Submission time	Start time	Running time	Avg. file throughput	Current job throughput
21.28 MiB	21.28 MiB	2021-03-22T09:47:23Z	2021-03-22T09:47:25Z (+2s)	2 s	35.46 MB/s	-

Showing 1 to 3 out of 3

SUBMITTED
DELETE
READY
STAGING
ACTIVE
STARTED
CANCELED
FAILED
3 FINISHED
NOT_USED

First
Previous
1
Next
Last

File ID	File State	File Size	Throughput	Remaining	Start Time	Finish Time	Staging Start	Staging End
+ 7288116	FINISHED	4.15 MiB	30.76 MB/s	-	2021-03-22T09:47:25Z	2021-03-22T09:47:27Z		
⬆ srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lofar/user/sksp/archive/NGLOW/logs/scheduler/2018-09-01.tar.gz								
⬆ srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lofar/user/sksp/distrib/distrib_natalie/2018-09-01_4.tar.gz								
+ 7288117	FINISHED	4.83 MiB	44.17 MB/s	-	2021-03-22T09:47:25Z	2021-03-22T09:47:27Z		
⬆ srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lofar/user/sksp/archive/NGLOW/logs/scheduler/2018-09-02.tar.gz								
⬆ srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lofar/user/sksp/distrib/distrib_natalie/2018-09-02.tar.gz								
+ 7288118	FINISHED	6.29 MiB	31.46 MB/s	-	2021-03-22T09:47:25Z	2021-03-22T09:47:27Z		
⬆ srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lofar/user/sksp/archive/NGLOW/logs/scheduler/2018-09-03.tar.gz								
⬆ srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lofar/user/sksp/distrib/distrib_natalie/2018-09-03.tar.gz								

At the moment any jobs are visible to anyone under any VO, but this can be closed by our system administrators upon request, just contact us at helpdesk@surfsara.nl.

Failed transfers

In case that you monitor any failed transfers, then once the bulk transfer finishes, collect them and resubmit only the list with the files that failed.

Make a list to retry the failed transfers:

```
$fts-transfer-status -s https://fts.grid.surfsara.nl:8446 --list [JOBID] | grep -3 ↵  
↵State:.*FAILED | egrep 'Source:|Destination:' | sed -e 's/ Source:      //' -e 'N;s/\n ↵  
↵Destination:/' > srm_fts_retry1.txt # replace the [JOBID] with your bulk job ID
```

Submit the failed transfers with:

```
$fts-transfer-submit -s https://fts.grid.surfsara.nl:8446 --retry 2 --retry-delay 300 --  
↵nostreams 4 --timeout 14400 -f srm_fts_retry1.txt >> fts_jobids
```

Using the API with curl

- Get user information:

```
$curl --capath /etc/grid-security/certificates --cacert /tmp/x509up_uXXXX --cert /tmp/  
↵x509up_uXXX https://fts.grid.surfsara.nl:8446/whoami  
##replace x509up_uXXXX with your proxy location and name
```

- Get your delegation ID:

```
$curl --capath /etc/grid-security/certificates --cacert /tmp/x509up_uXXXX --cert /tmp/  
↵x509up_uXXX https://fts.grid.surfsara.nl:8446/whoami | jq '.delegation_id'  
##3d3ce1e83def1abc
```

- Check the expiration time of our delegated credentials

```
$curl --capath /etc/grid-security/certificates --cacert /tmp/x509up_uXXXX --cert /tmp/  
↵x509up_uXXXX https://fts.grid.surfsara.nl:8446/delegation/3d3ce1e83def1abc  
##{"voms_attrs": ["/lsgrid/Role=NULL/Capability=NULL"], "termination_time": "2021-03-  
↵23T09:31:33"}  
##replace 3d3ce1e83def1abc with your delegation ID
```

- Get job information:

```
$curl --capath /etc/grid-security/certificates --cacert /tmp/x509up_uXXXX --cert /tmp/  
↵x509up_uXXXX https://fts.grid.surfsara.nl:8446/jobs/207773b6-8af8-11eb-90ed-  
↵fa163e7fa8c6  
##replace 207773b6-8af8-11eb-90ed-fa163e7fa8c6 with your job ID
```


globusonline client

This page includes the basic commands to use *globusonline*. For an overview of storage clients, see *Storage clients*.

Contents

- *globusonline client*
 - *About Globus Online*
 - *Storage Endpoints*
 - * *Personal endpoints*
 - *Activate a local machine endpoint*
 - *Activate Grid UI endpoint*
 - * *Server endpoints*
 - *Activate dCache endpoint*
 - *Data transfers*
 - * *dCache transfers*
 - *From local machine*
 - *From Grid UI*

About Globus Online

Globus Online provides a service to transfer files from one storage endpoint to another via the Web GUI or an API. The Globus Online web service allows to monitor the transfer processes via the web interface and reschedules transfers automatically when they fail.

There exists a python API which allows to steer and monitor the processes from within e.g. the data generating processes. Via this API data transfers can be easily integrated into workflows.

The service is run on the Amazon cloud located in the US.

Storage Endpoints

In order to access, share, transfer, or manage data using Globus, the first step is to create an endpoint on the system(s) where the data is (or will be) stored.

These endpoints can be:

- Globus Connect Personal endpoints:
 - local machines like your laptop or a Virtual Machine
 - personal accounts on login nodes connected to clusters like your home account on *SURF grid UI* or Spider UI.

Note: It is not possible to transfer data between two Personal endpoints, unless you use Globus Plus groups or managed subscriptions.


- Globus Connect Server endpoints:
 - gridFTP servers, like the *Grid Storage* or SURFsara's archive.

Personal endpoints

On servers or computers that do not run GridFTP you will first have to install the Globus Connect client software. Below we show how to define your laptop or your home folder on a User Interface machine as endpoint:

Activate a local machine endpoint

- Create an account [here](#) by following the instructions
- Login to the web interface:



Not Logged-In
[Home](#)

Log In with Globus ID

[Need a Globus ID? Sign Up](#)

The client **Globus Auth** is requesting access to your **globusid.org** account for accessing a third-party website or application located at **auth.globus.org**. If you approve, please log in to continue.

Username

natalieda

@globusid.org

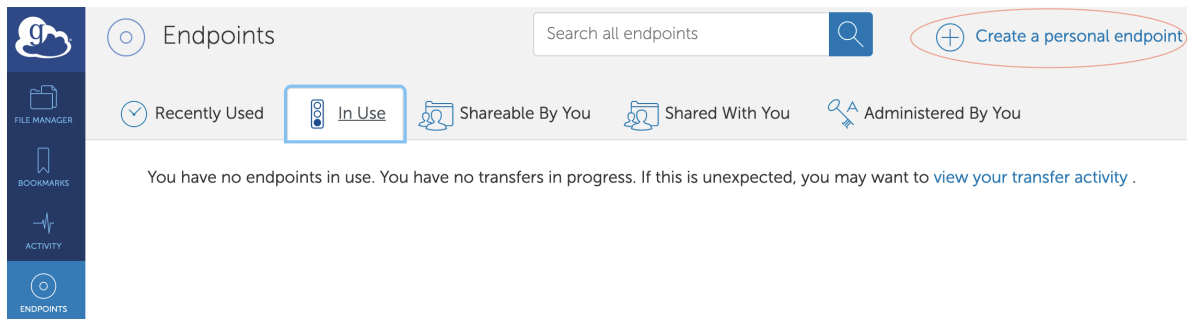
Password

.....

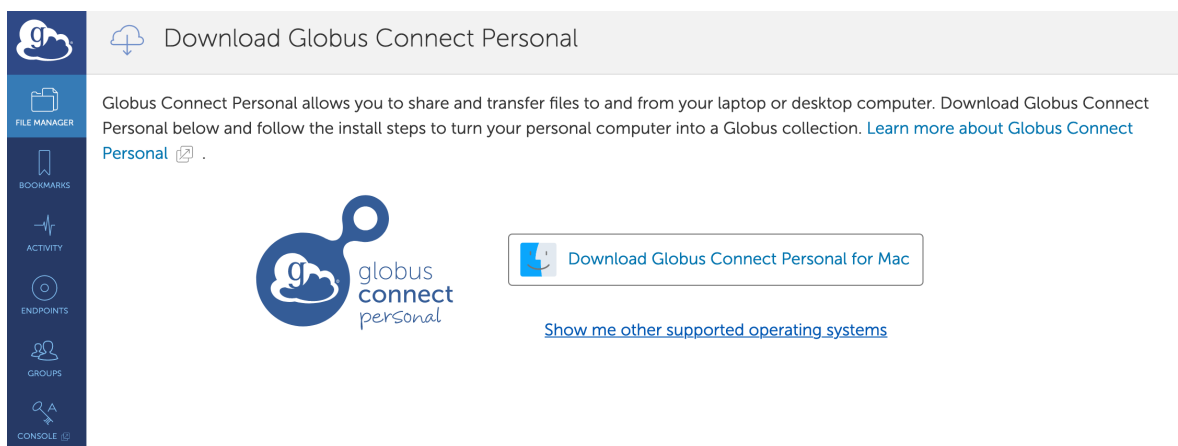
Log In

[Forgot password?](#)

- Install the Globus Connect client supported by the OS of your local machine as described [here](#). There are different ways to install the client. Here we show examples on a Mac OS local machine:
 - From the web interface select on the left ENDPOINTS and then click on '+Create a personal endpoint':





- Download and install the client:



Download Globus Connect Personal

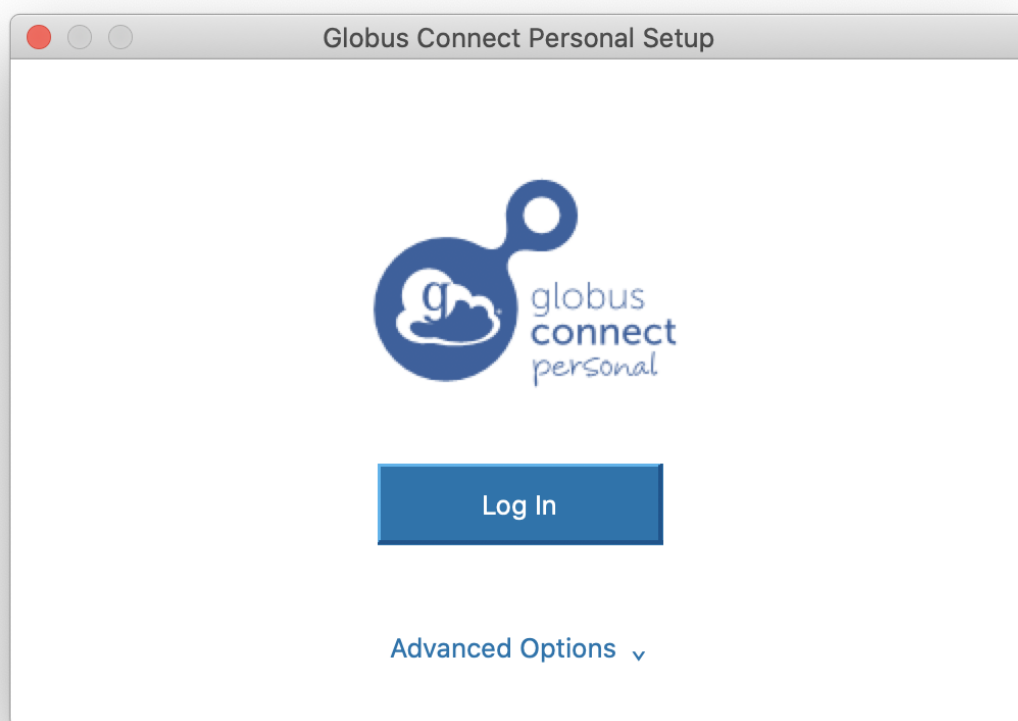
Globus Connect Personal allows you to share and transfer files to and from your laptop or desktop computer. Download Globus Connect Personal below and follow the install steps to turn your personal computer into a Globus collection. [Learn more about Globus Connect Personal](#).

 **globus connect personal**

 Download Globus Connect Personal for Mac

[Show me other supported operating systems](#)

- Start and setup the Globus Connect client:





Account ▾

Globus Connect Personal Setup would like to:

- ✓ View identity details ⓘ
- ✓ View your identity ⓘ
- ✓ View information about your linked identities ⓘ
- ✓ Create Globus Connect Personal collections in the Globus Transfer service ⓘ

Provide a label for future reference

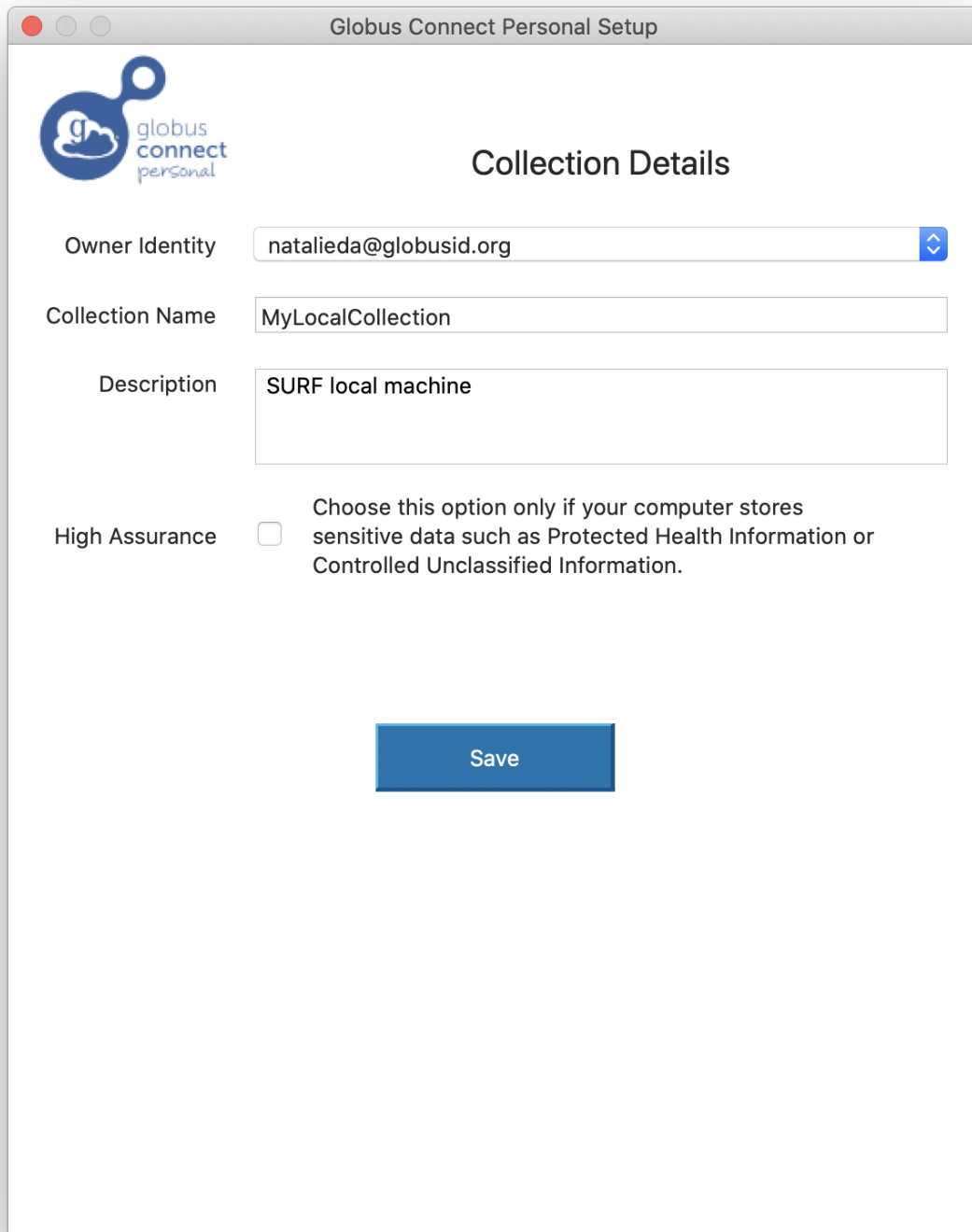
MyLocalMachine

You can rescind this consent at any time by visiting the [Manage Consents](#) ⓘ page.

By clicking "Allow", you allow **Globus Connect Personal Setup**, in accordance with its [terms of service](#) ⓘ and [privacy policy](#) ⓘ, to use the above listed information and services.


Allow

Deny



The screenshot shows a window titled "Globus Connect Personal Setup". In the top-left corner is the Globus Connect Personal logo. The main heading is "Collection Details". Below this, there are four fields: "Owner Identity" with the value "natalieda@globusid.org" and a dropdown arrow; "Collection Name" with the value "MyLocalCollection"; "Description" with the value "SURF local machine"; and "High Assurance" with an unchecked checkbox. To the right of the checkbox is a text block: "Choose this option only if your computer stores sensitive data such as Protected Health Information or Controlled Unclassified Information." At the bottom center is a blue "Save" button.

Globus Connect Personal Setup

 Collection Details

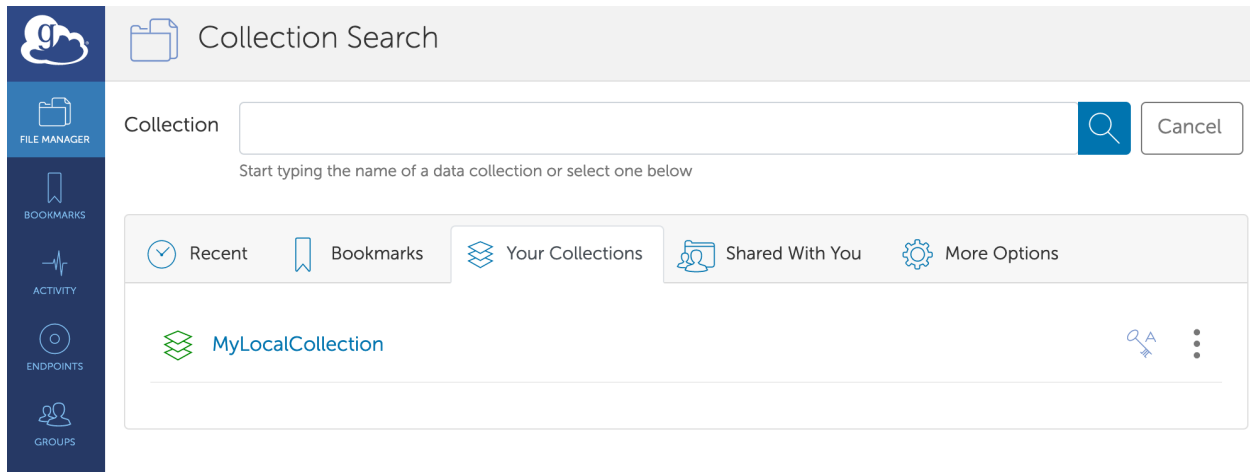
Owner Identity

Collection Name

Description

High Assurance ☐ Choose this option only if your computer stores sensitive data such as Protected Health Information or Controlled Unclassified Information.

- You should see the Globus Connect client running (e.g in the menu bar)
- Check in the web interface that your local machine is activated as Globus Online endpoint. On the left select FILE MANAGER, and in the collection search box find your collection and click on it. You should see your local files:



You can now use your local machine to transfer files to other Globus Online endpoints.

Activate Grid UI endpoint

For a linux machine as the Grid UI, you need to configure the linux package for Globus Connect Personal on your home UI account. The following steps assume that you already have a Globus Online account. If not, please refer to the previous section.

- Log in to your UI account:

```
$ssh -YC homer@ui.grid.surfsara.nl # replace "homer" with your username
```

- Download the linux package for Globus Connect Personal and extract the files:

```
$wget https://downloads.globus.org/globus-connect-personal/linux/stable/
→globusconnectpersonal-latest.tgz
$tar xzf globusconnectpersonal-latest.tgz
$cd globusconnectpersonal-x.y.z/ # replace x.y.z with the version number
```

- Start the installation and follow the steps. Execute the globus client on non graphical mode:

```
$/globusconnectpersonal -setup --no-gui
```

- A login url will be displayed. Copy and paste it in your browser
- Assuming that you already have a Globus account, login to globus using the displayed url
- Provide a label for future reference and click next. An authorization code will be displayed in your browser. Copy it and paste it on the Grid UI next to the prompt *Enter the auth code:*. Then provide a name for this collection as “Input a value for the Endpoint Name:”
- If everything went well then you should get a message “setup completed successfully”
- From now on you can use the Grid UI home account to transfer data via Globus. Everytime you use Globus, you need to start the client on the UI in order to activate it as an endpoint:

```
$/globusconnectpersonal -start
```

- The command above will define your home folder on a grid user interface machine as endpoint. If you wish to grant access to other paths that you have access on the same machine, then you can define a comma separated list of full paths that Globus may access as (If no prefix is present, r/w is assumed):

```
$ ./globusconnectpersonal -start -restrict-paths /project/myData/
```

Note: The `./globusconnectpersonal -start` command will keep your session open and the endpoint activated until you stop it with Ctrl+C

- Open a browser in your laptop (the Grid UI Firefox is quite slow) and login to your Globus account to see your new Grid UI Personal endpoint. It should be in status 'ready':

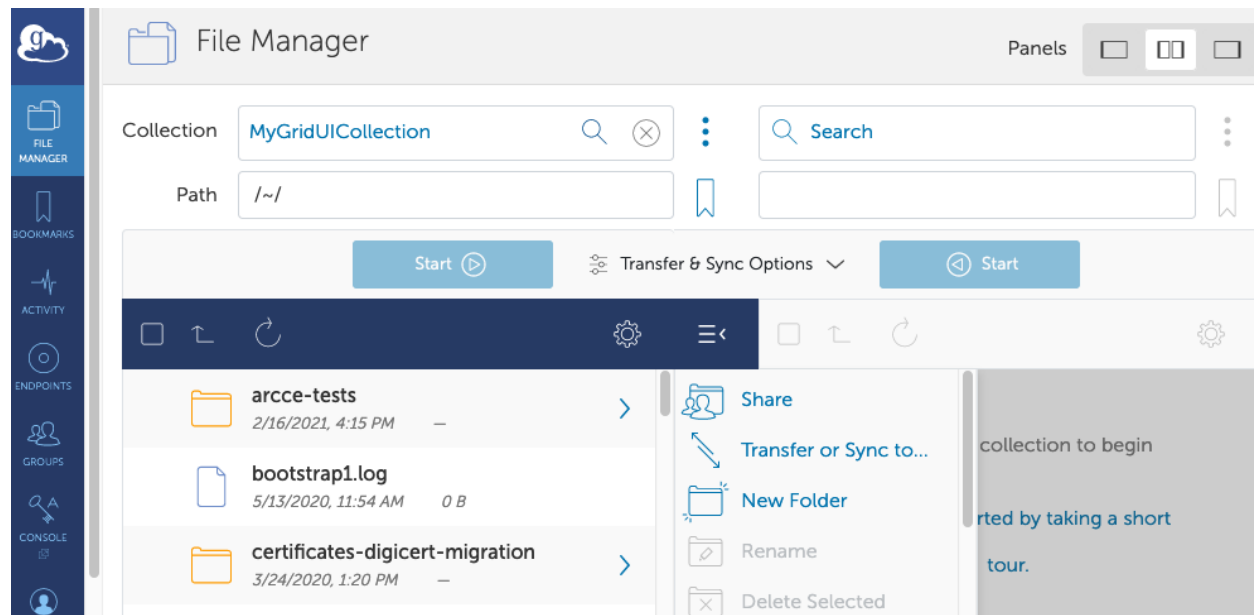
The screenshot shows the 'Endpoints' page in the Grid UI. The left sidebar contains navigation links: FILE MANAGER, BOOKMARKS, ACTIVITY, ENDPOINTS (selected), GROUPS, and CONSOLE. The main content area has a search bar 'Search all endpoints' and a '+ Create a personal endpoint' button. Below these are tabs: 'Recently Used', 'In Use', 'Shareable By You', 'Shared With You', and 'Administered By You' (selected). A filter 'Filter administered by you' is also present. The table below lists endpoints:

ENDPOINT	STRICT	STATUS	ROLE	SHARED
MyGridUICollection Globus Connect Personal		ready		
MyLocalCollection Globus Connect Personal		ready		

- Select the Grid UI endpoint from the tab 'Your Collections' in your FILE MANAGER to see your home data:

The screenshot shows the 'Collection Search' page in the Grid UI. The left sidebar is the same as the previous screenshot. The main content area has a search bar 'Collection' with a search icon and a 'Cancel' button. Below the search bar is the text 'Start typing the name of a data collection or select one below'. There are tabs: 'Recent', 'Bookmarks', 'Your Collections' (selected), 'Shared With You', and 'More Options'. The table below lists collections:

MyGridUICollection	
MyLocalCollection	



Note: To transfer data between two Globus Connect Personal endpoints, you must be a member of a Globus Plus Group or at least one of the endpoints must be managed under a Globus subscription (i.e Globus server endpoints).

Server endpoints

When you use GridFTP as transfer protocol either the source or the destination has to be GridFTP-enabled. The storage endpoints can be different servers or only different locations on the same server. Globus online makes use of Grid proxies for GridFTP transfers. The data transfer is executed on behalf of the user with the use of his/her Grid proxy. Thus, a user needs to have the following when using GlobusOnline together with dCache:

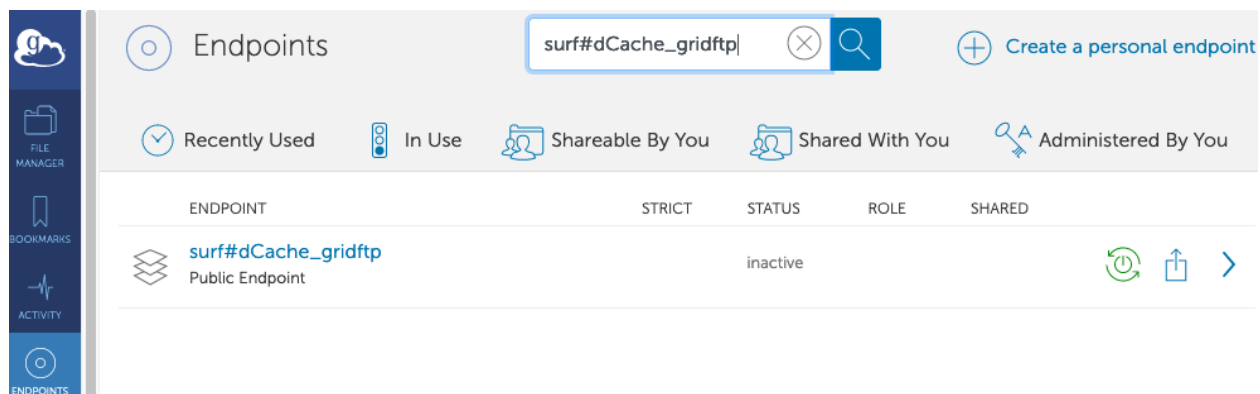
- A personal Grid certificate
- A VO membership

Activate dCache endpoint

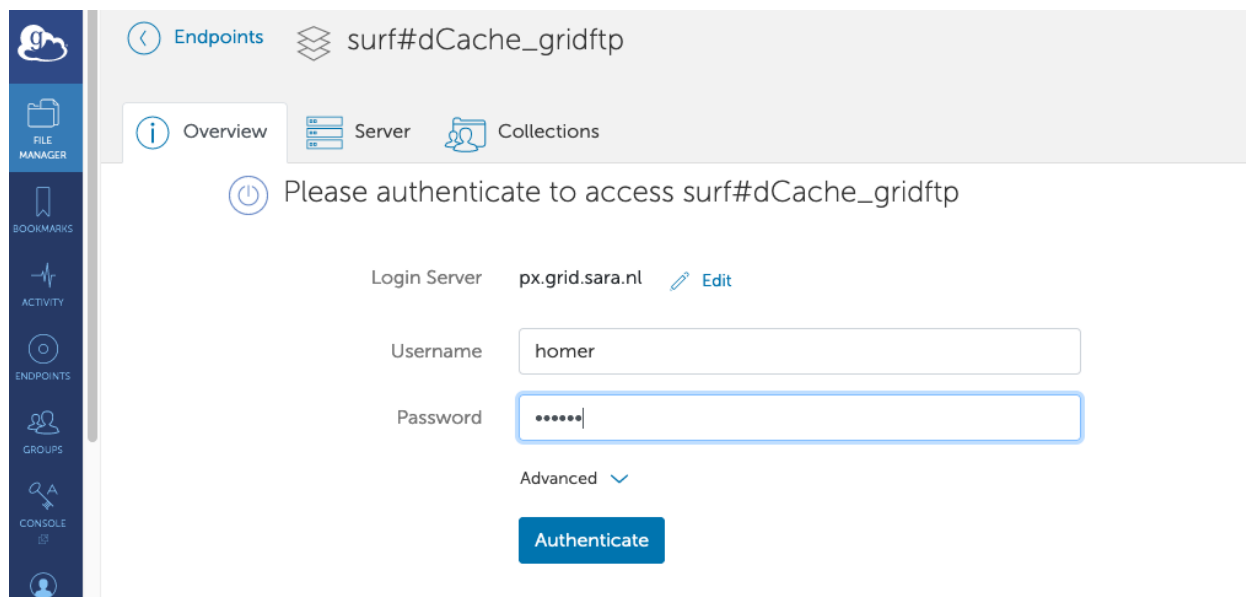
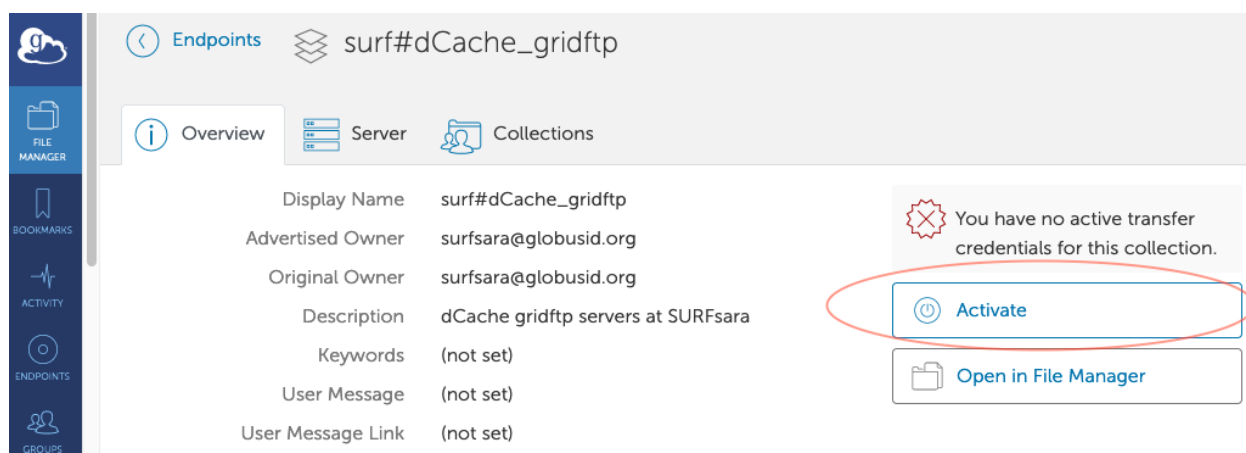
- To activate a GridFTP-enabled endpoint the user needs to provide the service with a Grid proxy. Login to the grid UI, start a Grid session and create a Grid proxy on the proxy server:

```
$ssh -YC homer@ui.grid.surfsara.nl
$startGridSession lsgrid #replace lsgrid with your VO
$myproxy-init --voms lsgrid -l homer #replace lsgrid with your VO and homer e.g. with_
→your name. The username is only valid for this proxy and could be anything you choose.
## Enter GRID pass phrase for this identity: [give your grid certificate password]
## Enter MyProxy pass phrase: [give any passphrase you like. This passphrase will be_
→used later to activate the dCache endpoint]
```

- Login to the Globus web interface from a browser in your laptop and on the left go to ENDPOINTS. On top search for surf#dCache_gridftp. This GridFTP endpoint points at the dCache Grid storage.



- To activate the dCache Gridftp endpoint, click on the `surf#dCache_gridftp` endpoint and then 'Activate'. Provide the username and passphrase from the previous step when you created a proxy on the grid UI and click 'Authenticate':



Endpoints **surf#dCache_gridftp**

Overview Server Collections

Display Name	surf#dCache_gridftp
Advertised Owner	surfsara@globusid.org
Original Owner	surfsara@globusid.org
Description	dCache gridftp servers at SURFsara
Keywords	(not set)
User Message	(not set)
User Message Link	(not set)
Endpoint Info Link	(not set)
Contact Email	(not set)
Organization	(not set)
Department	(not set)

⚠ Your activation expires in 12 hours (10:31am on Wednesday, March 17th, 2021)

⌂ Extend Activation

⏻ Deactivate

📁 Open in File Manager

- Check in the web interface that the dCache endpoint is activated as Globus Online endpoint. On the left select FILE MANAGER, and provide the following details in the 'Collection' and 'Path'. You should see your dCache VO files:

- Collection: surf#dCache_gridftp
- Path: /pnfs/grid.sara.nl/data/lsguid/SURFsara/ #replace with your VO and project path

Endpoints **surf#dCache_gridftp**

Overview Server Collections

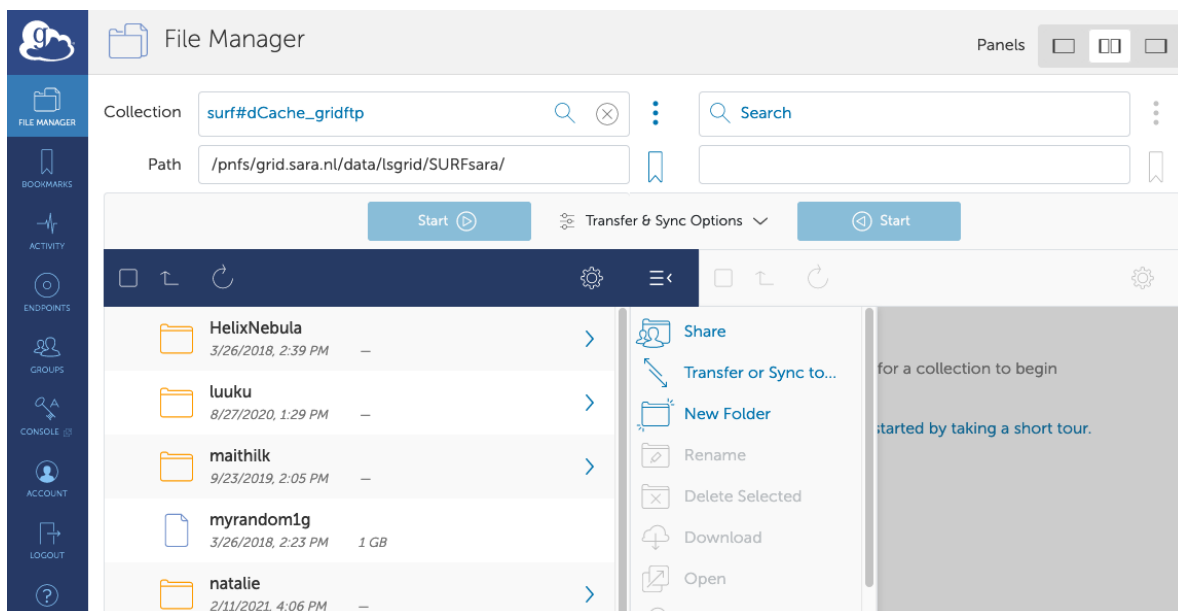
Display Name	surf#dCache_gridftp
Advertised Owner	surfsara@globusid.org
Original Owner	surfsara@globusid.org
Description	dCache gridftp servers at SURFsara
Keywords	(not set)
User Message	(not set)
User Message Link	(not set)
Endpoint Info Link	(not set)
Contact Email	(not set)
Organization	(not set)
Department	(not set)

⚠ Your activation expires in 12 hours (10:31am on Wednesday, March 17th, 2021)

⌂ Extend Activation

⏻ Deactivate

📁 Open in File Manager



Data transfers

Before data can be transferred you need to *activate the endpoints* from/to which data should be transferred. Globus Online executes data transfers on behalf of a user. Data transfers can be easily started employing the web interface. You have to provide the names of the endpoints from and to which the data is transferred.

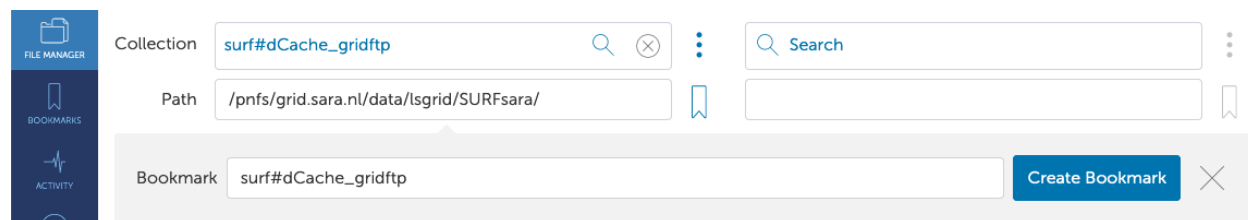
Data to be transferred is selected by marking it and then clicking one of the arrows to determine sink and source. The current state of data transfers can be monitored in the ACTIVITY tab.

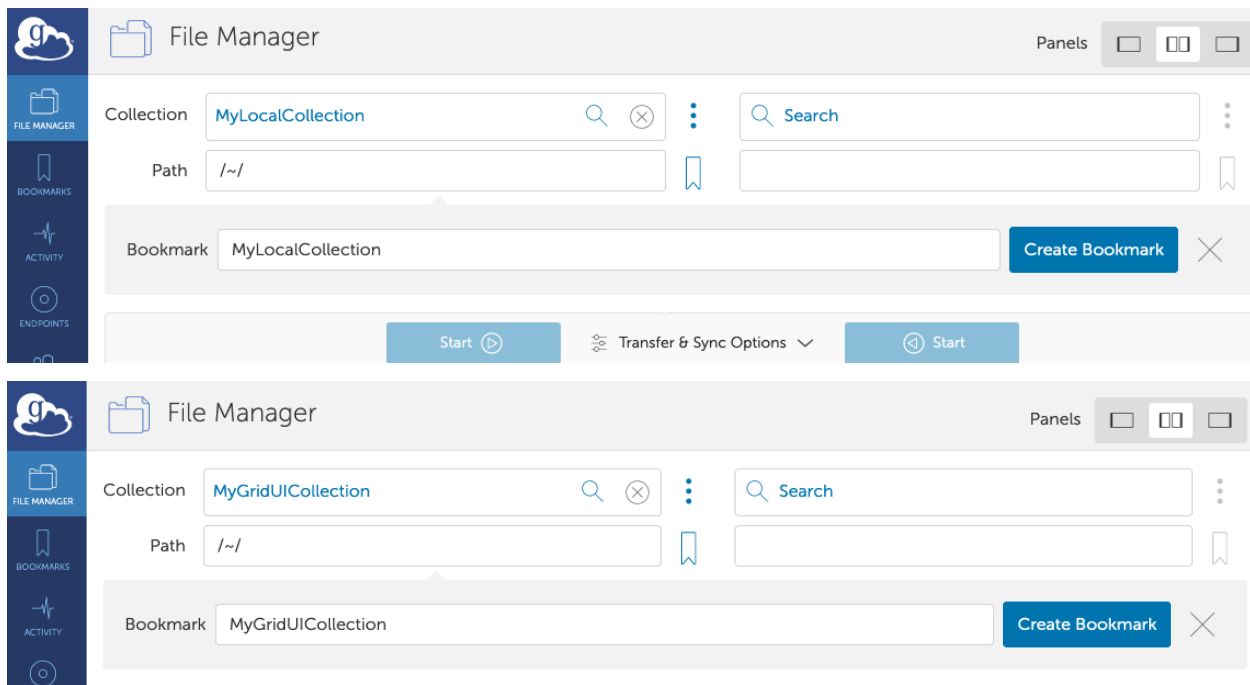
Interacting with Globusonline is possible via the Web Interface or with a python API. The examples here show the webinterface transfers.

dCache transfers

Transferring data to/from dCache requires that the dCache endpoint is activated and a Grid proxy is provided by the user and instantiated with his/her Grid credentials/certificates.

Creating bookmarks of your endpoints and desired storage paths allows you to define easily the source and target data locations prior to the transfers. You can create bookmarks on the Web interface:

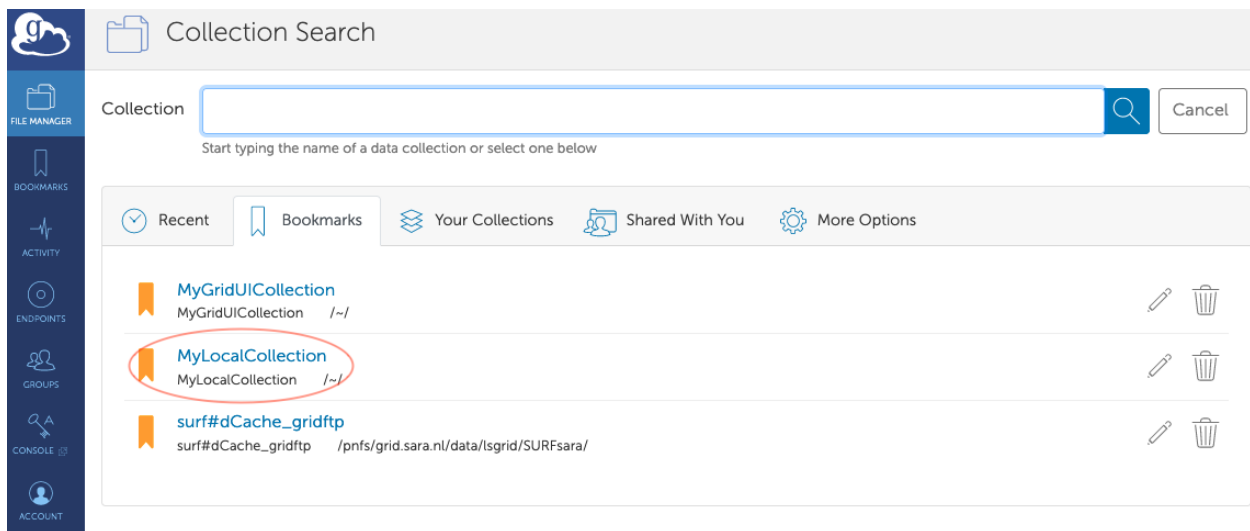


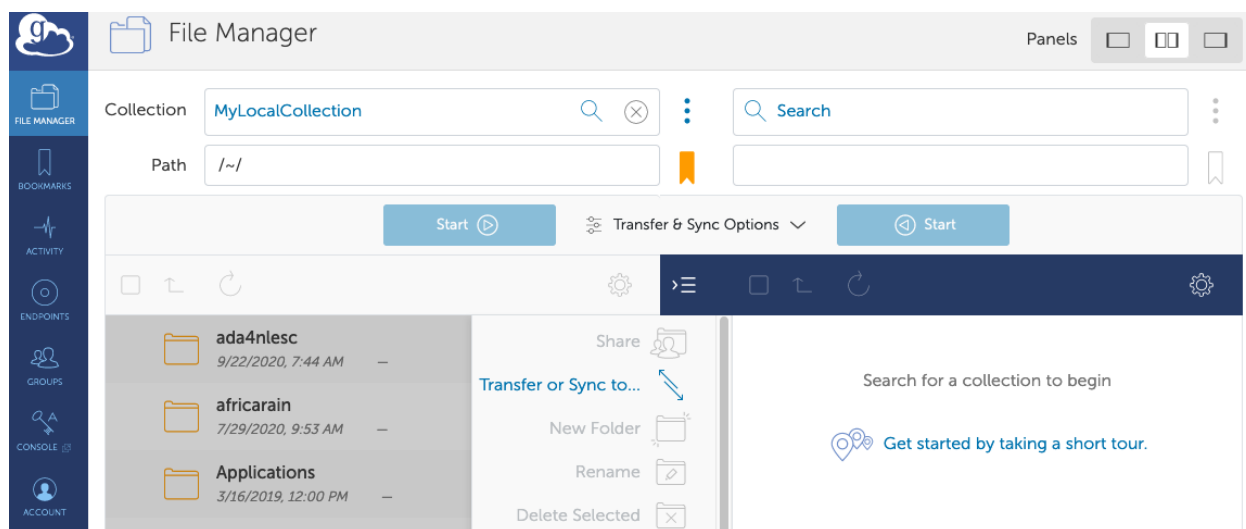


The next sections give some examples for data transfers between the activated endpoints.

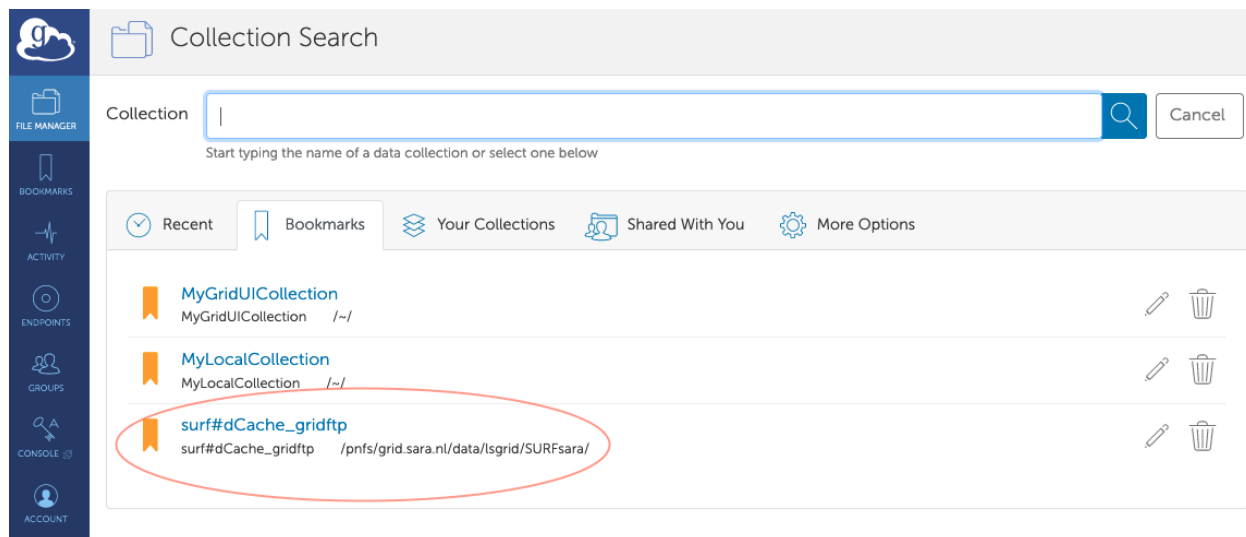
From local machine

- From the web interface, go to FILE MANAGER and click on the Collection box. Your bookmarks are displayed. Select your local machine:

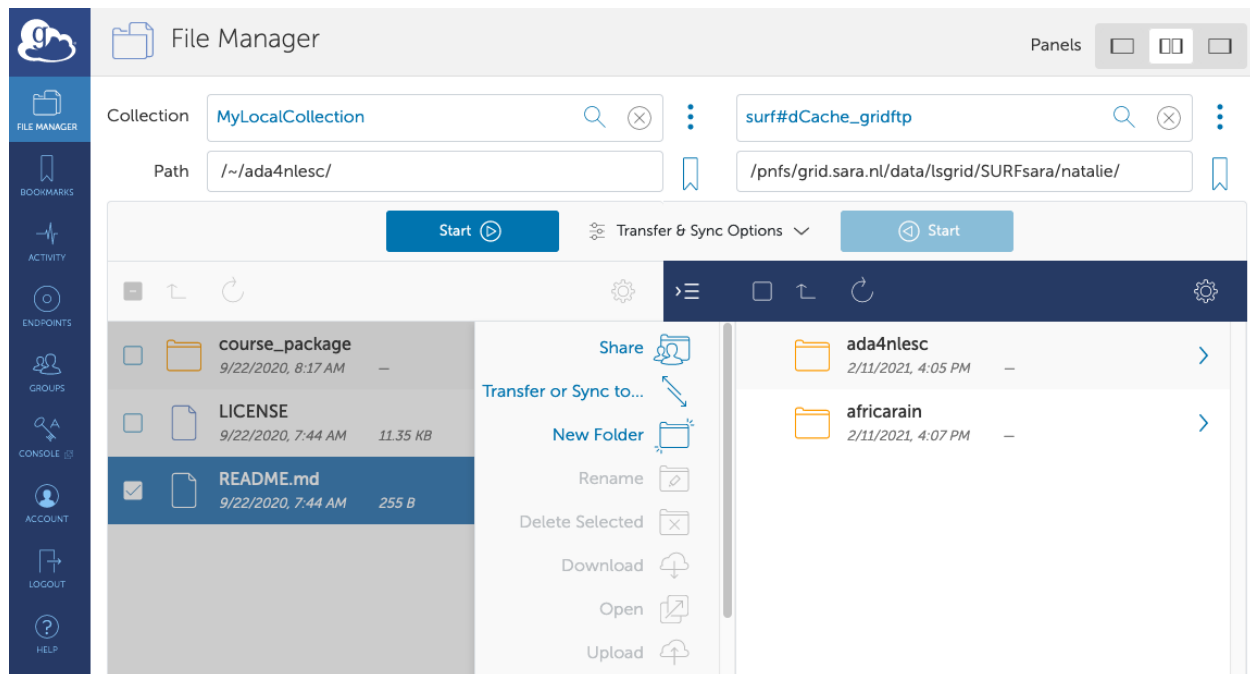




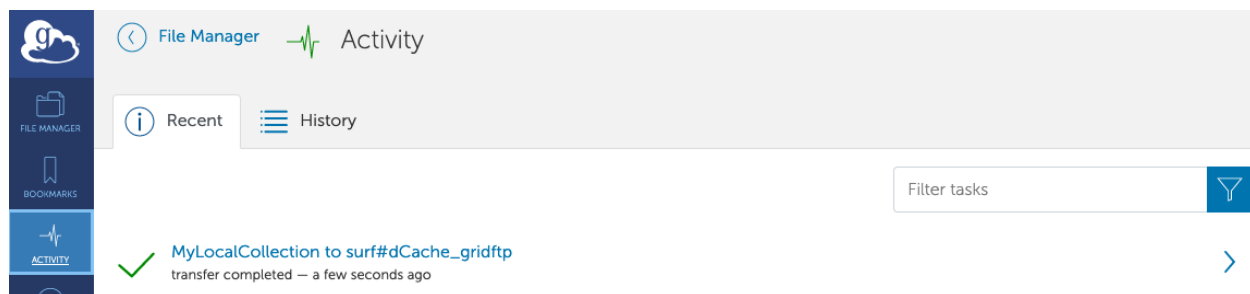
- Go to FILE MANAGER and on the right side of the page, click on the Collection box. Your bookmarks are displayed. Select your dCache_gridftp endpoint:



- Go to FILE MANAGER and select the source and target directories and files. When ready, click 'Start':

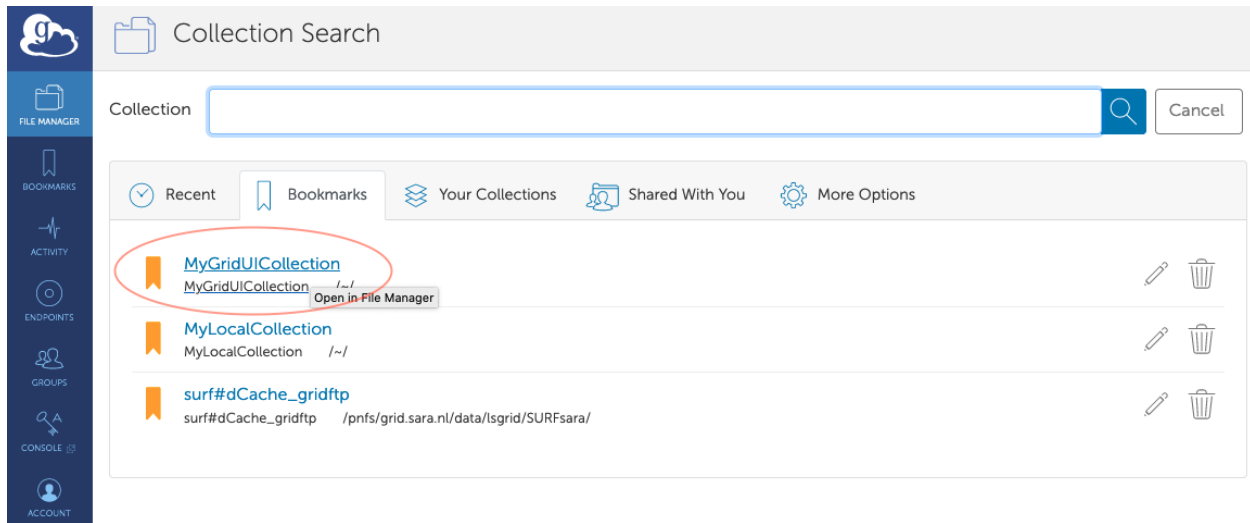


- In the ACTIVITY tab you can monitor the transfer progress:

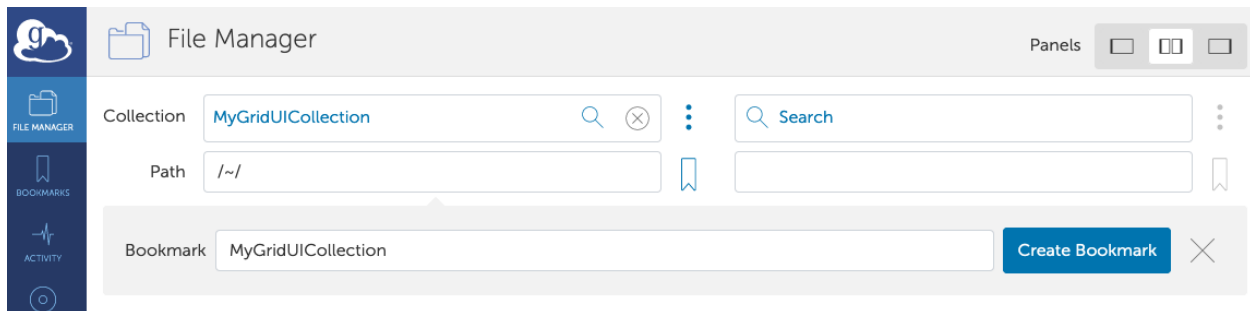


From Grid UI

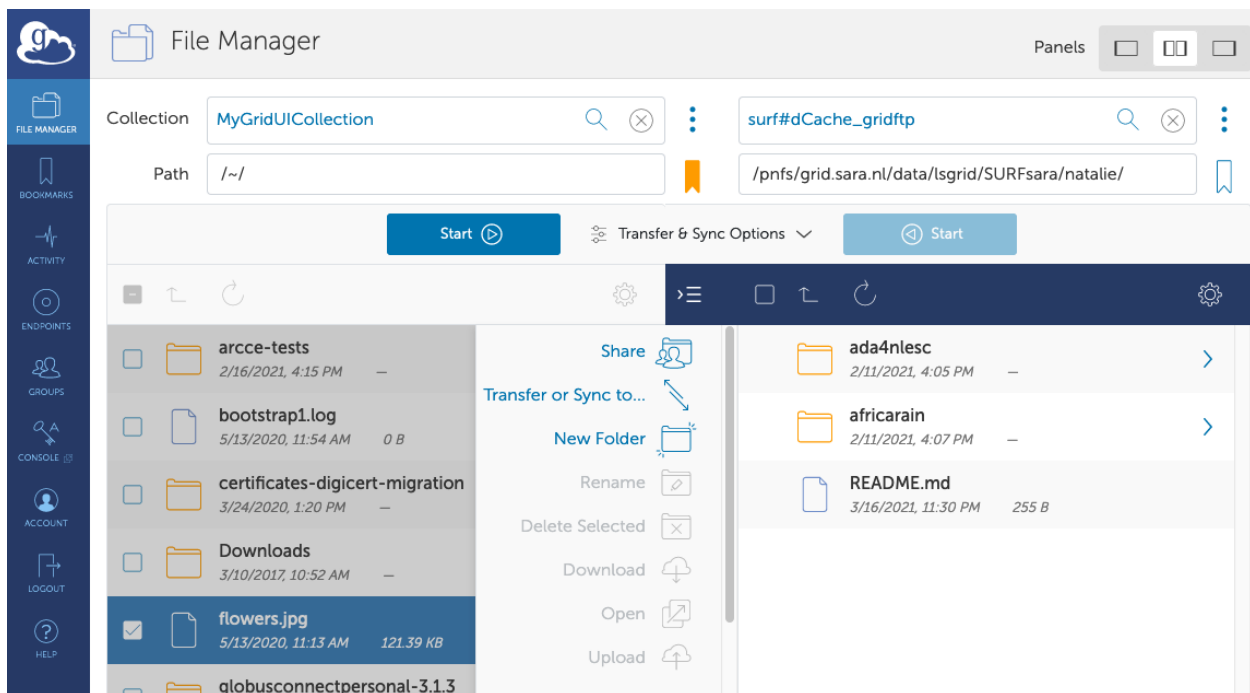
- From the web interface, go to FILE MANAGER and click on the Collection box. Your bookmarks are displayed. Select your Grid UI endpoint:



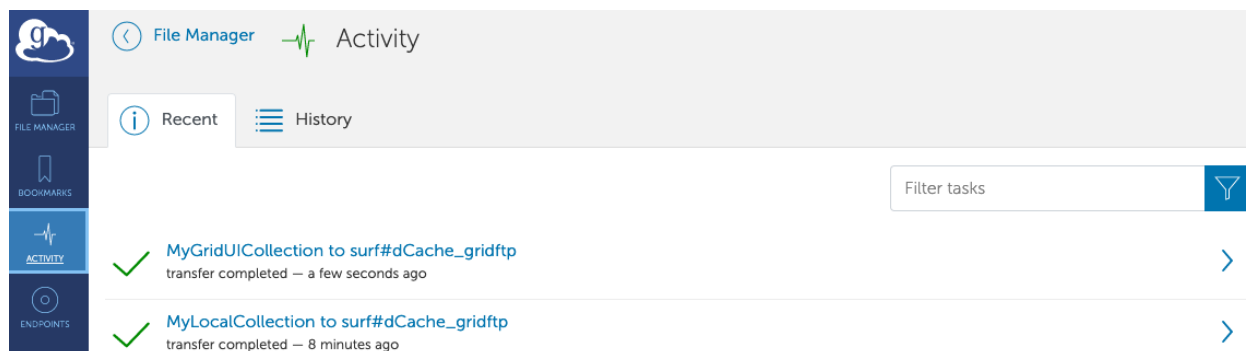
- Go to FILE MANAGER and on the right side of the page, click on the Collection box. Your bookmarks are displayed. Select your dCache_gridftp endpoint:



- Go to FILE MANAGER and select the source and target directories and files. When ready, click 'Start':



- In the ACTIVITY tab you can monitor the transfer progress:



Refresh the directory contents of the target endpoint to see your transferred files.

3.2.5 Staging files

The dCache storage at SURFsara consists of magnetic tape storage and hard disk storage. If your *quota allocation* includes tape storage, then the data stored on magnetic tape has to be copied to a hard drive before it can be used. This action is called *Staging files* or ‘bringing a file online’.

Note: Staging is important. If your job reads a file that is on tape but not online, your job will wait until dCache brings the file online (or reaches a timeout). This may take minutes when it’s quiet, but it may take days when multiple users are staging large datasets. That would be a waste of CPU cycles. But that’s not all: the number of concurrent transfers is limited per pool, so it would also be a waste of transfers slots.

Table 2: Staging terms

Locality	Meaning
ONLINE	The file is only on disk
NEARLINE	The file is only on tape; it should be staged before reading it
ONLINE_AND_NEARLINE	The file is both on disk and on tape

There are some more file statuses. See the [SRMv2 specifications](#) for a full list.

The Grid storage files remain online as long as there is free space on the disk pools. When a pool group is full (maximum of assigned quota on staging area) and free space is needed, dCache will purge the least recently used cached files. The tape replica will remain on tape.

The amount of time that a file is requested to stay on disk is called `pin lifetime`. The file will not be purged until the `pin lifetime` has expired.

There are different ways to stage your files on dCache. For running the staging commands, it is required to authenticate to dCache either with a valid proxy certificate or username/password or a macaroon depending on the client. We support the following clients:

- [gfal2](#) supported by CERN. It allows proxy authentication only.
- [ada](#) supported by SURF. It allows proxy or username/password or macaroon authentication.
- [srmbringonline](#). It supports proxy authentication only.

The preferred method for bulk staging is the [gfal2](#) practice below.

gfal2 python API

The gfal2 API provides a set of functions to support staging operations. Given a SURL or TURL list of files, the following gfal2 scripts can be used:

- `state.py`: display the locality (state) of the files
- `stage.py`: stage (bring online) the files and return a token
- `release.py`: release the files based on the token

The following section gives an example of usage of the gfal2 scripts.

Preparation

- Download the scripts on the UI and inspect the usage instructions inside each of `state.py`, `stage.py`, `release.py`.
- Create a file that includes a list with SURLS (`srm://srm.grid.sara.nl/..`) of the files you want to stage from tape, e.g. the file called `mysurls` inside the folder `datasets`:

```
$cat datasets/mysurls
#srm://srm.grid.sara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/file1
#srm://srm.grid.sara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/file2
#srm://srm.grid.sara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/file3
```

- In case that you plan to stage a big bulk of data, split the filelist in chunks of 1000 files:

```
$split -l 1000 --numeric-suffixes [datasets/mysurls] [output_prefix]
```

State operations

- Check the state of the files with:

```
$python state.py --file [datasets/mysurls]
#srm://srm.grid.sara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/file1 ONLINE_AND_NEARLINE
#srm://srm.grid.sara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/file2 NEARLINE
#srm://srm.grid.sara.nl/pnfs/grid.sara.nl/data/lsgrid/homer/file3 NEARLINE
```

If your surls filelist is too long it is better to redirect the output of this command in a file.

- Display the total number of files on tape and/or disk:

```
$python state.py --file [datasets/mysurls] | awk '{print $2}' | sort | uniq --count
#1 ONLINE_AND_NEARLINE
#2 NEARLINE
```

- If your surls filelist is too long, please use the token returned by the `stage.py` (see section below) to retrieve the state of your files. It improves performance significantly! Given a token id [5d43ee2e:-1992583391] the equivalent commands would be:

```
$python state.py --file [datasets/mysurls] --token [5d43ee2e:-1992583391]
$python state.py --file [datasets/mysurls] --token [5d43ee2e:-1992583391] | awk '{print
→$2}' | sort | uniq --count
```

Stage operations

- Submit a staging request for your surls filelist. The command returns a token that you can use to check the state of the files:

```
$python stage.py --file [datasets/mysurls]
#Got token 5d43ee2e:-1992583391
```

You can store the output in a file stage.log to make sure that you keep safe the token IDs of your stage requests. This is required to state/release the files later on.

Note: The token always corresponds to the given surls file list. You need to use the exact same surls file in your state/release commands. If you add/remove any file in your surls filelist and use the old token, you will get an error.

Note: In case that you plan to stage a big bulk of data, then **submit each chunk of 1000 files with 1 minute interval time** to prevent overloading the staging namespace server.

- The pin lifetime is set in seconds, the default pin time in the script is set to two weeks (or 1209600 sec). You can request the desired pin time with the argument `[-pintime pintime]`:

```
$python stage.py --file [datasets/mysurls] --pintime [pintime in seconds]
#Got token 5d43ee2e:-1992583393
```

The pin lifetime counts from the moment you submit the request independent to the actual time that the files are on disk.

Unpinning operations

- Release the pin of your bulk of files with:

```
$python release.py --file [datasets/mysurls] --token [5d43ee2e:-1992583391]
```

After submitting the unpinning command above, the files will remain cached but purgeable until new requests will claim the available space.

If your surls filelist is too long it is better to redirect the output of this command in a file.

Monitor staging activity

Once you submit your stage requests, you can use the gfal scripts to monitor the status or check the webpage below that lists all the current staging requests:

<http://dcmain.grid.sara.nl:2288/poolInfo/restoreHandler/lazy>

Unpin a file

The disk pool where your files are staged has limited capacity and is only meant for data that a user wants to process. When you stage a file you set a `pin lifetime`. The pin lifetime here is set to 1 week, but note that this counts from the moment you submit the request independent to the actual time that the files are on disk. The file will not be purged until the pin lifetime has expired. Then the data may be purged from disk, as soon as the space is required for new stage requests. When the disk copy has been purged, it has to be staged again in order to be processed on a Worker Node.

When a pool group is full with pinned files, staging is paused. Stage requests will just wait until pin lifetimes for other files expire. dCache will then use the released space to stage more files until the pool group is full again. When this takes too long, stage requests will time out. So pinning should be used moderately.

When you are done with your processing, we recommend you release (or unpin) all the files that you don't need any more. It is an optional action, but helps a lot with the effective system usage.

- Release the pin of your bulk of files with:

```
$python release.py --file [datasets/mysurls] --token [5d43ee2e:-1992583391]
```

This command will initiate unpinning of files in `mysurls` list. If your `surls` filelist is too long it is better to redirect the output of this command in a file.

Warning: At the moment neither the `srm-bring-online` nor the `python gfal release.py` scripts can effectively release a file if there are multiple pin requests. Please use `srm-release-files` if you want to release all the pins set to a file.

3.2.6 Checksums

dCache checks the checksum of a file on many operations (for instance, during tape store & restore operations). If dCache finds, that the checksum of a file does not match the checksum it has in its database, dCache will refuse to continue and will present an error message instead.

dCache is configured to use Adler32 checksums by default, for performance reasons. It is however possible to transfer files to dCache while verifying MD5 checksums. Globus Online works only with MD5 checksums, and previous versions of dCache did not support MD5 checksums, so one would have to disable checksum checking in Globus. Now, dCache does support MD5 checksums during transfers, even when the default checksum type is Adler32. So now Globus Online and dCache should be able to work together with checksums enabled. If a GridFTP client uploads data with MD5 verification enabled, dCache will calculate the MD5 checksum, return this to the client and store it in its database.

dCache does not enable a user to add MD5 checksums of **existing** data.

We may however, if your project needs it, change the default checksum from Adler32 to MD5 for your poolgroups. From the moment we do that, for new files, dCache will store MD5 checksums in its database, and this MD5 checksum will be used to verify file integrity during operations. For existing data, we can tell dCache to calculate the new checksums. Tape data will have to be staged to do this. MD5 checksums require more CPU than Adler32, so there will be a performance impact for writing, staging and reading data.

Checksums can be listed with `srm ls -l`:

```
$srm ls -l srm://srm.grid.sara.nl/pnfs/grid.sara.nl/data/lsgrid/test | grep locality
```

Also through [webdav clients](#) and [gfal client](#) you can retrieve a file's checksum. The checksum value comes from the database so it performs well.

3.2.7 Transport security

With most protocols, the authentication is done over a secured channel (usually encrypted with TLS). After authentication, the data is transferred in plain text. Since most users are from High Energy Physics, this is usually no problem.

Life scientists however may need to store private data such as MRI scans or DNA scans. The European GDPR law requires careful handling of such personal data. We therefore suggest that you keep such data safe during transport. Below we give an overview which protocols in dCache encrypt data during transfer.

Data channel encryption:

- WebDAV over port 2880, 2881, 2883 and 2884 (we configured 2881 and 2884 to have the most secure encryption)
- WebDAV over port 443 only through <https://webdav-cert.grid.sara.nl>

NO data channel encryption:

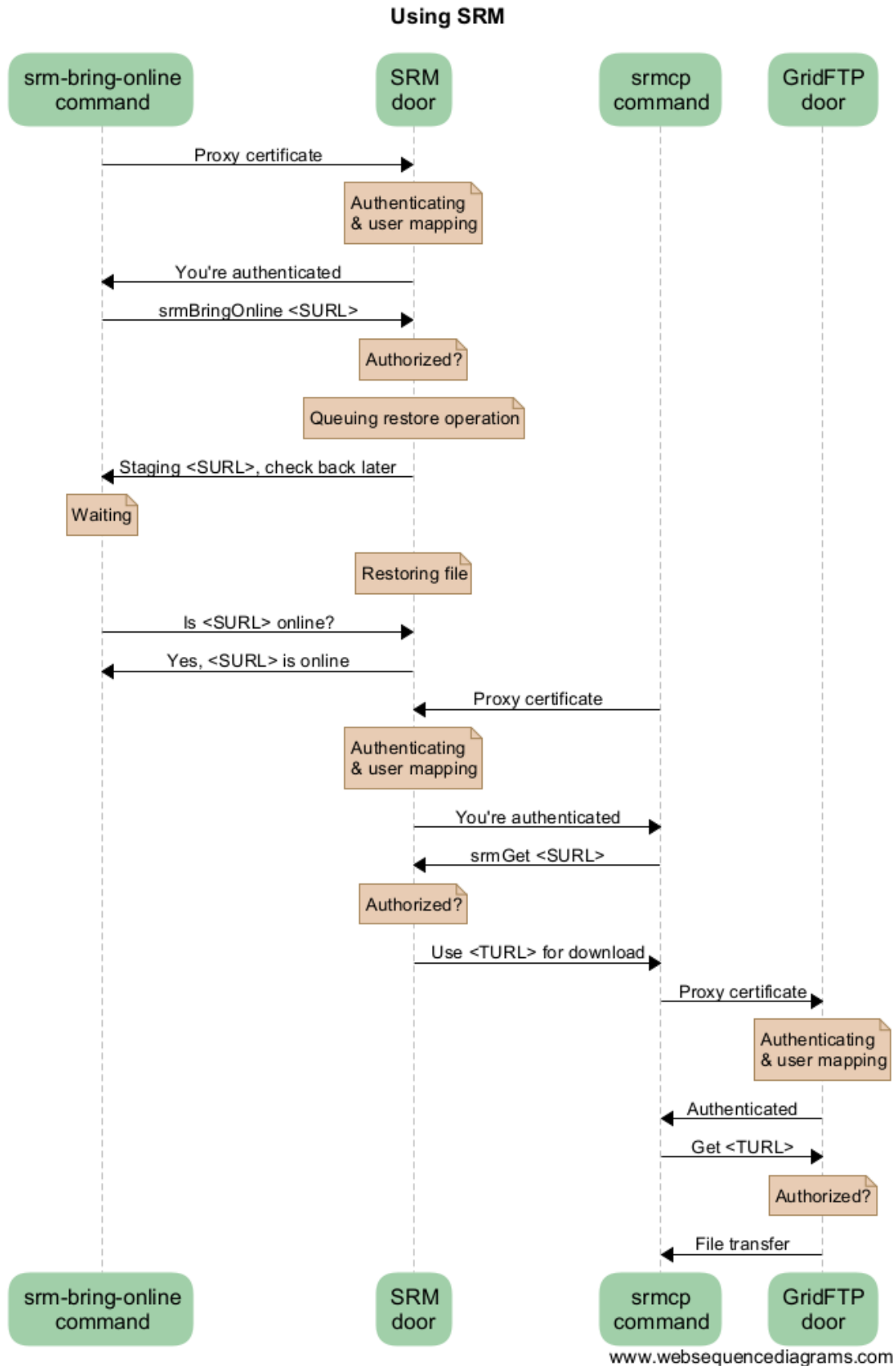
- WebDAV over port 443
- WebDAV over port 2882
- SRM
- GridFTP (dCache does not support GridFTP data encryption. Please be warned that `globus-url-copy -dcprv` does not warn you about this and transfers your data in plain text.)
- GSIdCap, dCap
- Xroot

Since WebDAV is currently the only way to encrypt data in transit, we continuously try to improve the security of our WebDAV doors. We regularly test our WebDAV doors with tools like [the Qualys SSLtest](#), [nmap](#), [Greenbone/OpenVAS](#), and others, and follow their recommendations.

The conclusion: if your data is personal, safely upload it to and download it from dCache, by using WebDAV over ports 2881 or 2884. See [webdav clients](#) for more information.

3.2.8 SRM interaction example diagram

Here is a sequence diagram that illustrates how the SRM commands interact with the Grid storage.



As you can see from this diagram, there can be a lot of overhead per file. For this reason, Grid storage performs best with large files. We recommend to store files of several megabytes to gigabytes. Some users have files of more than 2 terabytes, but that may be impractical on other systems with limited space like worker nodes.

3.2.9 Importing large amounts of data

The [Data Ingest Service](#) is a SURFsara service for researchers who want to store or analyse large amounts of data at SURFsara. The service is convenient for users who lack sufficient bandwidth or who have stored their data on a number of external hard disks.

3.3 Grid job requirements

By telling what your job needs, you help the scheduler in finding the right place to run your jobs, and it also helps using the different compute nodes in the Grid efficiently.

This chapter describes how to write the requirements, how the requirements determine where your jobs will run, and what they tell the scheduler. Job requirements are provided as parameters in the JDL file.

Contents

- *Grid job requirements*
 - *Requirement syntax*
 - *Requirements*
 - * *Specifying Wall Clock time via the queue*
 - * *Specifying CPU Time*
 - * *Selecting particular Grid site or CE*
 - * *Multicore jobs*

3.3.1 Requirement syntax

Job requirements are written as an optional statement in the JDL file:

```
Attribute = <parameter>;
```

See also:

For detailed information about JDL attributes supported by the DIRAC, have a look in the [DIRAC documentation](#).

3.3.2 Requirements

Specifying Wall Clock time via the queue

Typically on grid sites, by specifying the wall clock time requirement the scheduler picks a queue which is long enough for running the job. In the DIRAC configuration at SURF site (Gina), only the long queue (Walltime of 96 hours) is supported at this point. You can use the following convention to specify it:

```
Tags={"long"};
```

If you do not specify this, the jobs that run at SURF will automatically be scheduled on the long queue. If you specify any other queue your jobs will not run at SURF site.

Specifying CPU Time

You may also choose to specify a CPU Time, but at SURF site all jobs will land on the long queue (96 hours) irrespective of the CPU Time specified. But specifying this attribute may offer flexibility for you to use the same JDL for different grid sites where a CPU Time attribute is used to match the job to the correct queue. You can specify this attribute as follows:

```
CPUTime = 345600; #this is in seconds for the long queue
```

Warning: If a job is not actively using CPU for more than 30 min, it will be considered as stalled and automatically killed by DIRAC Watchdog

Selecting particular Grid site or CE

You may choose a specific Grid site or a CE to run your jobs on, depending on which sites allow jobs for the specific VO:

```
Site = {"GRID.SURF.nl"};
GridCE = {"arc01.gina.surfsara.nl"};
```

If you are using DIRAC to submit your jobs, you do not have to specify either one of the parameters (unless you want your jobs to/not to run at a specific site).

Multicore jobs

NumberOf Processors is the number of CPU cores requested. SMPGranularity is the number of cores that must be scheduled on the same host:

```
# Request just 4 cores on a single node
SmpGranularity = 4;
NumberOfProcessors = 4;
```

Note that if you do not specify SmpGranularity the requested number of cores can be distributed over different nodes, which is only useful for MPI (or likewise) applications.

Warning: If you are running a multi-core process in your job, and you do not set the correct number of CPU cores, you will oversubscribe a compute node, slowing down your own analysis, as well as others.

3.4 Grid certificates

In this section we discuss Grid user certificates in more detail; how to convert certificates, and how to find out the details of your Grid certificate.

Contents

- *Grid certificates*
 - *Certificate and key file inspection*
 - * *Using the modulus to see whether a key and a certificate match*
 - * *Finding the expiry date of your certificate*
 - * *Finding the subject of your certificate*
 - *Conversion of key and certificate formats*
 - * *Converting from PKCS12 to PEM*
 - * *Converting from PEM to PKCS12*

Once you have obtained and installed a *legacy certificate* there are some `openssl` commands that can help you inspect the information stored in your certificate. This section will show you how to do this.

3.4.1 Certificate and key file inspection

Sometimes you want to view the details of your key and/or your certificate file. Details like:

- do the key and the certificate file go together?
- when does the certificate expire?
- what is the subject or DN (Distinguished Name) of the certificate?

Using the `openssl` command, you can find out the details of your certificates and keys.

Using the modulus to see whether a key and a certificate match

The modulus is a short message that can be used to identify if a private key and certificate match. If they do not match, the private key and certificate are useless.

To find the modulus of your key, use:

```
openssl rsa -in userkey.pem -noout -modulus | openssl md5
```

You will be asked to provide the password you used to protect your key file. To find the modulus of your certificate, use:

```
openssl x509 -in usercert.pem -noout -modulus | openssl md5
```


If the md5 sum of the moduli of the key file and the certificate file do not match, you cannot use that combination to identify yourself.

Finding the expiry date of your certificate

To find out when your certificate is valid, use:

```
openssl x509 -in usercert.pem -noout -dates
```

This will tell you when your certificate is valid.

Note that a key does not have a validity period.

Finding the subject of your certificate

The subject or DN of a certificate is the human-readable identification of who the certificate belongs to. It usually contains your name, country, organisation and your e-mail address.

To find out who the certificate belongs to, use:

```
openssl x509 -in usercert.pem -noout -subject
```

3.4.2 Conversion of key and certificate formats

Private keys and certificates can be stored in different formats. Different systems use different formats. The two important formats are:

- PEM: stores keys and certificates in separate ascii-files; this format is used by the Grid middleware and storage programs;
- PKCS12: stores keys and certificates in one binary file; this format is used by browsers.

Sectigo creates PKCS12 files, whereas *DutchGrid* creates PEM files.

Converting from PKCS12 to PEM

To convert a PKCS12 file to the PEM format, you need two commands; one to extract the key, and one to extract your certificate.

To extract your key, run:

```
openssl pkcs12 -in browsercert.p12 -out userkey.pem -nocerts
```

Note that you will first need to enter the password that was used to *create* the PKCS12 file. Next, you need to enter a password to protect the exported key. Enter that password again to verify. Note that you must enter a password and the password must be at least 12 characters and include non-alphanumerics; if the password is too short, `openssl` will fail without error. You can use the same password as for the PKCS12 file.

To extract your certificate, run:

```
openssl pkcs12 -in browsercert.p12 -out usercert.pem -nokeys -clcerts
```

Converting from PEM to PKCS12

To convert your certificate in PEM format to the PKCS12-format, use:

```
openssl pkcs12 -export -inkey userkey.pem -in usercert.pem -out browsercert.p12
```

This will ask you for a password three times: the first is to unlock your private key stored in the file `userkey.pem`. The PKCS12-file will be password protected, which needs a new password, and the same password for confirmation. Note that you can use the same password as for the private key, but this is not required.

When you import the PKCS12-file into your browser or keychain, you need to enter the password you used to protect the PKCS12-file.

Contents

- *Grid host certificates*
 - *Certificate Revocation Lists*
 - *Telling your browser to trust host certificates*

3.5 Grid host certificates

Apart from Grid user certificates, there are also Grid host certificates. Host certificates are used to establish trust between hosts on the grid. If you use our user interfaces or Grid compute clusters, you don't have to worry about these; we'll make sure they contain the right host certificates and their dependencies.

If you configure your own user interface, or if you intend to run GridFTP or Webdav clients on your own machines, you will need to install the Grid Certificate Authority (CA) root certificates. Your client software needs CA root certificates to know whether to trust the host certificate of the server they're talking to.

If you want to install the Grid CA root certificates, please follow the instructions here: https://wiki.egi.eu/wiki/EGI_IGTF_Release

After installation, you will find the Grid CA root certificates in `/etc/grid-security/certificates/`. You can point clients like `curl` there:

```
$curl --capath /etc/grid-security/certificates/ ....
```

Note: Many clients allow you to bypass verification of host certificates. This may seem like an easy solution, but it bypasses a security layer. We recommend that you don't bypass host certificate checking but install the Grid CA root certificates.

3.5.1 Certificate Revocation Lists

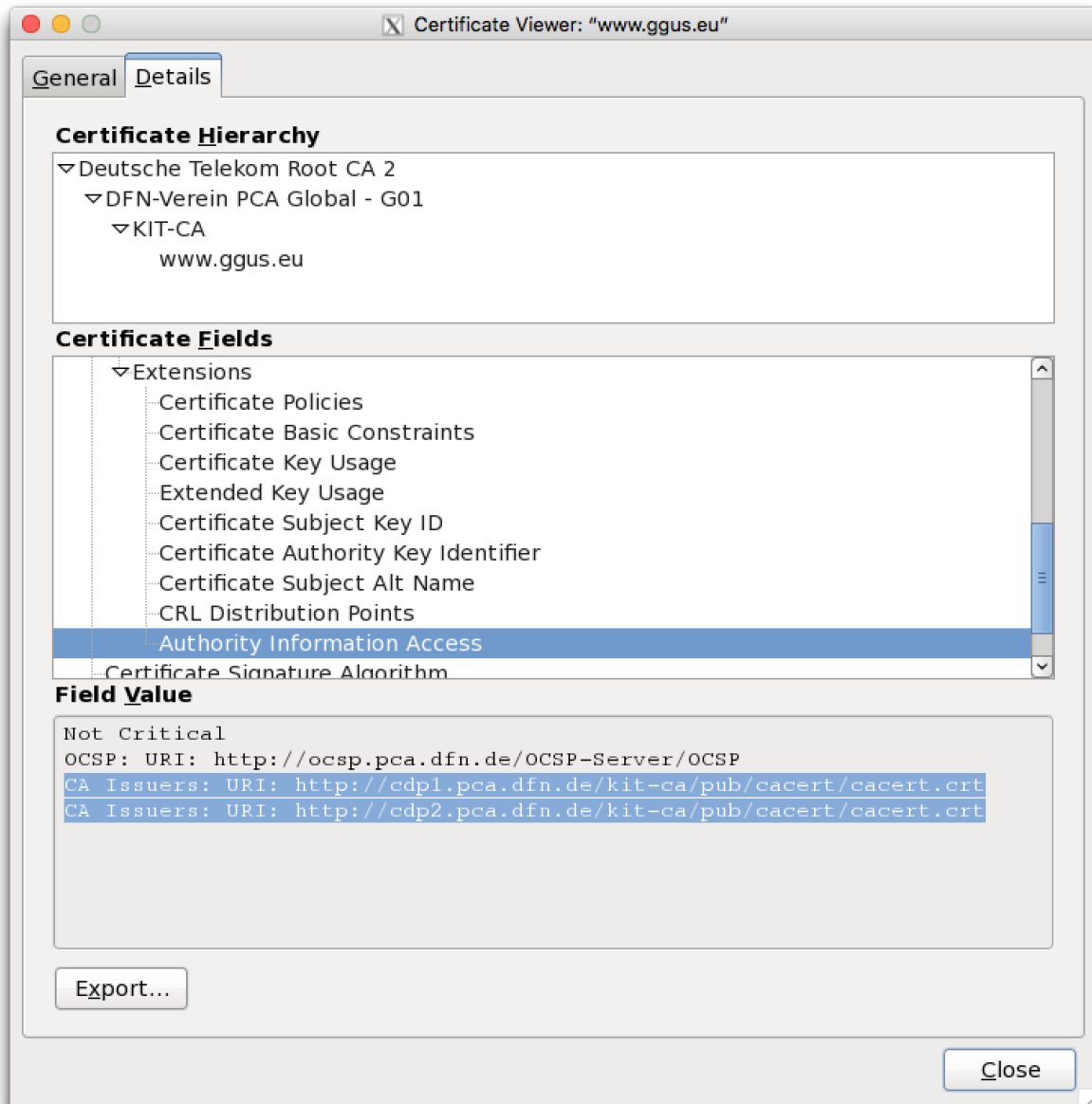
Certificate Revocation Lists or CRLs declare which host certificates have been revoked, for instance because they have been compromised. A utility called `fetch-crl` downloads these CRLs. It's good practice to set up `fetch-crl` as a cron job. You can find more information about this at <https://wiki.nikhef.nl/grid/FetchCRL3>.

3.5.2 Telling your browser to trust host certificates

Not every browser trusts Grid host certificates. You might see the error `SEC_ERROR_UNKNOWN_ISSUER` in Firefox. This means, that Firefox does not have the root CA certificate that was used to sign a specific host certificate. The solution is, to import the root CA certificate into the browser. But which one? Here are some tips to help you.

The [eugridpma.org](https://www.eugridpma.org) website has a map that helps you to find relevant Certificate Authorities and their root CA certificates: <https://www.eugridpma.org/members/worldmap/>

In Firefox, you can view the certificate information of a website. Very often, this information includes a link to the issuing CA root certificate you need to trust this certificate. See this image:



Here is a command to get the issuer of a host certificate, in this example of `webdav.grid.surfsara.nl`:

```
$echo | openssl s_client -connect webdav.grid.surfsara.nl:443 -CApath /etc/grid-security/
certificates/ 2>/dev/null \
| openssl x509 -noout -text \
| egrep -o 'https{0,1}://.*\.(pem|crt|der)'
```

This will show:

```
http://cacerts.digicert.com/TERENAEScienceSSLCA3.crt
```

You can copy this link and paste it in the URL field to import the root certificate into Firefox. After that, Firefox will trust all sites with host certificates signed by this root certificate.

BEST PRACTICES

4.1 Parametric jobs

The *pilot jobs* use the technique of parametric jobs for the job submission to the Grid. In this page we give an example of parametric jobs:

Contents

- *Parametric jobs*
 - *About*
 - *Example*

4.1.1 About

Pilot jobs are submitted to the Grid with a specific *Job Description Language* type called **Parametric**. A parametric job causes a set of jobs to be generated from one JDL file.

4.1.2 Example

In the example below, the parametric job will create 3 child jobs that will all run the same executable. The value %j will be replaced by the actual value of Parameters during the JDL expansion.

ParameterStart defines the starting value for the variation, **ParameterStep** the step for each variation and **Parameters** defines the value where the submission of jobs will stop (that value itself is not used) . The number of jobs is: $(\text{Parameters} - \text{ParameterStart}) / \text{ParameterStep}$

- Log in to your User Interface.
- Create a file with the following content describing the job requirements. Save it as `parametric.jdl`:

```
1 [
2   Type = "Job";
3   JobName = "Parametric";
4   ParameterStart = 0;
5   ParameterStep = 1;
6   Parameters = 3;
7
8   Executable = "/bin/hostname";
```

(continues on next page)

(continued from previous page)

```

9  Arguments = "-f";
10  StdOutput = "StdOut_%j";           #Name of the file to get the
   ↪ standard output stream
11  StdError = "StdErr_%j";           #Name of the file to get the
   ↪ standard error stream
12  OutputSandbox = {"StdOut_%j", "StdErr_%j"};  #j placeholder replaced by Dirac
   ↪ job ID
13
14  Site = "GRID.SURF.nl";             #Job destination site
15  NumberOfProcessors = 1;           #Number of cores on a single node,
   ↪ options: 1,2,4, or 8
16  Tags = {"long"};                 #Queue name, long default walltime
   ↪ is 96 hours
17  #CPUTime = 1000;                  #Max CPU time required by the job
   ↪ in seconds
18 ]

```

- You can submit the parametric job as any Grid job:

```
$dirac-wms-job-submit parametric.jdl -f jobIds
```

In this case, 3 child jobs will be generated. Each job will generate two files: StdOut_0 and StdErr_0, StdOut_1 and StdErr_1, StdOut_2 and StdErr_2.

- Monitor the job status to see the the parent job and the 3 child jobs with their status:

```

$dirac-wms-job-status -f jobIds
JobID=82 Status=Running; Site=GRID.SURF.nl; MinorStatus=Running;
JobID=83 Status=Running; Site=GRID.NIKHEF.nl; MinorStatus=Running;
JobID=84 Status=Running; Site=GRID.SURF.nl; MinorStatus=Running;
JobID=85 Status=Running; Site=GRID.NIKHEF.nl; MinorStatus=Running;

```

This is just an example. In practice you shouldn't send more than **50** jobs this way (Parameters=50). The parametric jobs is the technology used for submitting the pilot jobs. There is no need to monitor their status or retrieve the job output through Dirac as the *pilot frameworks* will take care of this.

4.2 Bootstrap application

When you have a binary program that you want to execute on the Grid you need to create a bootstrap application. This will execute a wrapper script that contains all the necessary information for your job to run. In this page we will show an example to run your bootstrap application on the Grid:

Contents

- *Bootstrap application*
 - *Problem description*
 - *Quickstart example*
 - * *Preamble*
 - * *Run locally*

* *Run on the Grid*

4.2.1 Problem description

You want to execute your own binary program on the Grid. When you have a binary program that you want to execute on the Grid you can send it along with the job submission. This can be done when the executable is not too large. The limit is about 100MB, for larger executables you can use *Softdrive* or the *Grid storage*.

Bootstrap basics

See also:

Have a look at our mooc video *Executables on Grid* for a simple example to get started.

To send such an executable along we will use the InputSandBox in the job description. The program itself will be executed by a simple shell script ("wrapper.sh"). There are several reasons to wrap the call to your executable with a script. One important one is that the executable file might not have executable permissions after it is copied to the Grid worker node. A second is that it is more flexible in the use of input parameters and also to redirect the output. In short, this script provides the correct environment for the execution of the binary.

In this page we will demonstrate a simple bootstrap application using the fractals example.

4.2.2 Quickstart example

Preamble

- Log in to the User Interface (UI):

```
$ssh homer@ui.grid.sara.nl # replace homer with your username
```

- Copy the tarball bootstrap_fractals-dirac.tar to your UI directory.
- Copy the fractals source code fractals.c to your UI directory.
- Untar the example and check the files:

```
$tar -xvf bootstrap_fractals-dirac.tar
$cd bootstrap_fractals-dirac/
$mv ../fractals.c ./
$ls -l

-rw-r--r-- 1 homer homer fractals.c
-rw-rw-r-- 1 homer homer fractals.jdl
-rw-rw-r-- 1 homer homer wrapper.sh
```

- Compile the example:

```
$cc fractals.c -o fractals -lm
```

Warning: It is advisable to compile your programs on the User Interface (UI) Machine. The Grid nodes have similar environments and the chance of your job to run successfully on a remote worker node is larger when your program is able to run on the UI.

Run locally

- Run the example locally on the UI with a set of parameters to understand the program:

```
$. ./fractals -o output -q 0.184 -d 2280 -m 4400 # try different parameters, e.g. -q 0.184 -d 2280 -m 4400
```

This will take a while, depending on the input parameters you selected. Once finished, it will create the “output” file.

- Convert the output file to .png format and display the picture:

```
$convert output "output.png"
$display output.png
```

Run on the Grid

- Create a proxy valid for a week:

```
$source /etc/dirac/pro/bashrc #enable the software environment to use Dirac tools
$dirac-proxy-init -b 2048 -g pvier_user -M pvier --valid 168:00 # replace pvier with your VO
```

- Inspect the JDL file `fractals.jdl`:

```
[
Type = "Job";
Jobname = "bootstrap";
Executable = "/bin/sh";
Arguments = "wrapper.sh";
StdOutput = "stdout";
StdError = "stderr";
InputSandbox = {"wrapper.sh", "fractals"};
OutputSandbox = {"stdout", "stderr", "output"};
]
```

In the JDL file we specify the content of the in- and output sandboxes. These sandboxes allow you to transfer small files to or from the Grid. The input sandbox contains all the files that you want to send with your job to the worker node, like e.g. the `fractals` script that you want executed. The output sandbox contains all the files that you want to have transferred back to the UI, e.g. the output `fractals` image.

- Inspect the contents of the `wrapper.sh` script:

```
$cat wrapper.sh
#!/bin/bash
chmod u+x fractals
./fractals -o output -q 0.184 -d 2280 -m 4400
...
```


Once this jobs lands on the Grid, it will execute the `wrapper.sh` script which is a master script to set the program environment and initiate the program execution. In the `wrapper.sh` script you may include also the commands to retrieve input from a Grid storage location or transfer the output results to a Grid storage location.

- Submit the job to the Grid:

```
$dirac-wms-job-submit fractals.jdl -f jobIds
```

- Check the job status from command line on the UI:

```
$dirac-wms-job-status -f jobIDs
```

- Once the job is finished, get the job output to the UI:

```
$dirac-wms-job-get-output -f jobIds
```

- Convert the output file to .png format and display the picture:

```
$convert output-dir/output "output.png" # replace with your job output directory
$display output.png
```

4.3 Picas Overview

This page is about the PiCaS pilot framework:

Contents

- *Picas Overview*
 - *About PiCaS*
 - * *Source Code*
 - *Picas server*
 - *Picas client*
 - * *How to use*
 - *Grant access on Picas*
 - *Background: CouchDB*
 - * *Picas views*

4.3.1 About PiCaS

Let's say that you have a number of *tasks* to be processed on the Grid Worker Nodes. Each task requires a set of parameters to run. The parameters of each task (run id, input files, etc) construct an individual piece of work, called *token*, which is just a description - not the task itself.

The central repository for the tasks is PiCaS, a Token Pool Server. The pilot jobs request the next free token for processing from PiCaS. As said, the content of the tokens is opaque to PiCaS and can be anything your application needs.

PiCaS works as a queue, providing a mechanism to step through the work one token at a time. It is also a pilot job system, indicating that the client communicates with the PiCaS server to fetch work, instead of having that work specified in a jdl (or similar) file.

The server is based on [CouchDB](#) (see [CouchDB book](#)), a NoSQL database, and the client side is written in Python.

Source Code

The source code of PiCaS is on [Github Picas Client](#).

4.3.2 Picas server

More about Picas Server?

See also:

Check out our mooc videos Picas server side [Part I](#) and [Part II](#).

On the server side we have a queue which keeps track of which tokens are available, locked or done. This allows clients to easily retrieve new pieces of work and allows also easy monitoring of the resources. As every application needs different parameters, the framework has a flexible data structure that allows users to save different types of data. Because of this, tokens can contain any kind of information about a task: (a description of) the work that needs to be done, output, logs, a progress indicator etc.

The server is a regular CouchDB server with some specific views installed. For more information on this, see the *Background: CouchDB* section.

4.3.3 Picas client

More about Picas Client?

See also:

Check out our mooc video [Picas client side](#)

The PiCaS client library was created to ease communication with the CouchDB back-end. It allows users to easily upload, fetch and modify tokens. The system has been implemented as a Python Iterator, which means that the application is one large for loop that keeps on running as long as there is work to be done. The client is written in Python and uses the python couchdb module, which requires at least python version 2.6. The client library is constructed using a number of modules and classes, most important of these are:

- The *Actors* module contains the `RunActor` class. This is the class that has to be overwritten to include the code that calls the different applications (tasks) to run.
- The *Clients* module contains a `CouchClient` class that creates a connection to CouchDB.
- The *Iterators* module contains classes responsible for working through the available tokens. The `BasicViewIterator` class assumes that all the token information is encoded directly into the CouchDB documents.
- The *Modifiers* module is responsible for modification of the PiCaS tokens. It makes sure the tokens are modified in a uniform way.

How to use

The source code includes a simple example of how to use PiCaS. You'll have to write your pilot job in Python. Using PiCaS boils down to these steps:

1. Extend the `RunActor` class and overwrite the necessary methods:
 - `process_token`; should be implemented and contain the code to process your task.
 - `prepare_env`; is called only once, before the first run of this pilot job. For example, this is a good place to download a database that will be used by all tasks.
 - `prepare_run`; is called before each task. For example, this is a good place to create output directories or open database connections
 - `cleanup_run`; is called after each task. For example, this is a good place to clean up anything setup in `prepare_run`. Note that it is not guaranteed that this code will be run. For example in the case of a failure during `process_token`
 - `cleanup_env`; is called only once, just before the pilot job is shutting down. For example, this is a good place to clean up anything setup during `prepare_env` (remember to clean up your temporary files when you leave!). Note that it is not guaranteed that this code will run as the job can be interrupted or crash before this.
2. Instantiate a `CouchClient`
3. Instantiate a `BasicTokenModifier`
4. Instantiate a `BasicViewIterator` (and provide it the `CouchClient` and `BasicTokenModifier`)
5. Instantiate your `RunActor` subclass (and provide it the `BasicViewIterator` and `BasicTokenModifier`)
6. Call the `run()` method of your `RunActor` to start processing tokens

For another example, see the `picas-example` in our documentation.

4.3.4 Grant access on Picas

Any user with a Grid project and allocated *quotas* can get a PiCaS account and also obtain a database on the CouchDB server. If you want access, just contact us at helpdesk@surfsara.nl to discuss your design implementation and request your PiCaS credentials.

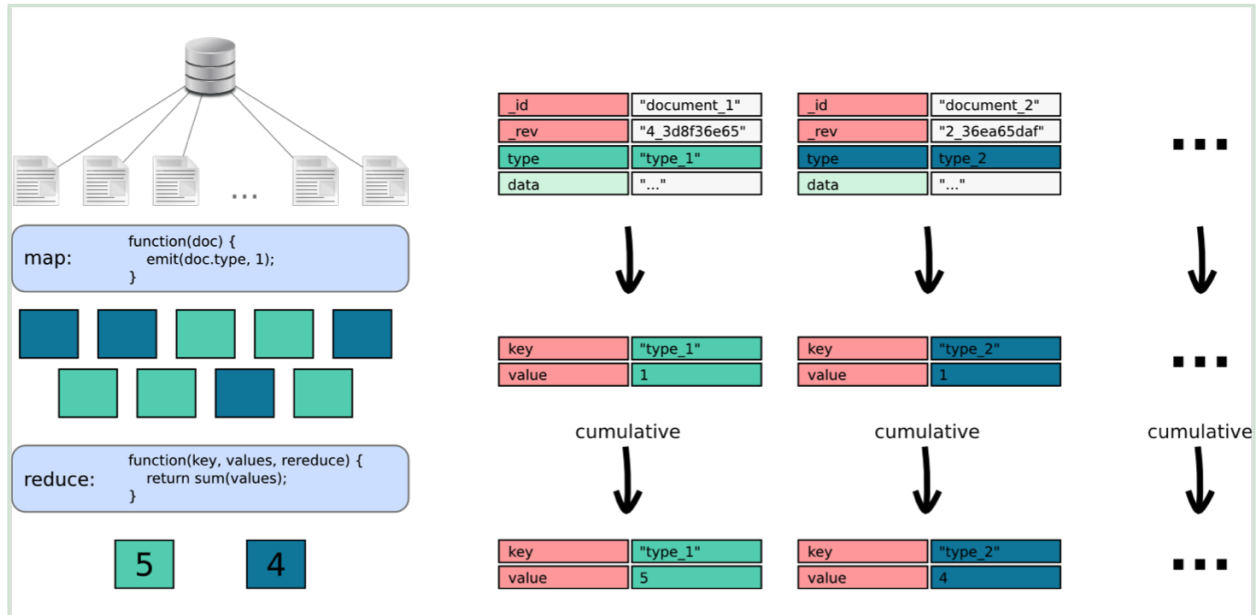
4.3.5 Background: CouchDB

PiCaS server is based on CouchDB. CouchDB stores documents which are self-contained pieces of information. These documents support a dynamic data model, so unlike traditional databases, CouchDB allows storing and retrieving any piece of information as long as it can be defined as key-value pairs. This feature is used to store all the information needed to keep track of the job stages and all of the required in- and outputs.

CouchDB also provides a Restful HTTP API, which means that we can easily access information with an HTTP client. This can be a browser, a command-line application like `curl` or a complete client library. It is also possible to interact with the CouchDB database behind PiCaS using the web-interface.

Picas views

CouchDB views are the basic query mechanism in CouchDB and allow you to extract, transform and combine data from different documents stored in the same database. This process is based on the Map/Reduce paradigm. In the case of CouchDB, the Map step takes every document from a database and applies a piece of code. It then sorts the output of that step based on the key that you supply and give it to the reducer. The code you supply for the reducer combines data from the mapper that have the same key.



The map code works on a 'per document' basis, so every document is run through that code one by one. The emit statement returns the value to the reduce command, again, this is all done for every document. In this case we are only interested in the type of the document, and as we want to count how many of each type there are, we provide the type as the key for the emit statement.

4.4 Picas Example

This page presents a PiCaS pilot job example:

Contents

- *Picas Example*
 - *Problem description*
 - * *Prerequisites*
 - *Picas sample example*
 - * *Prepare your Tokens*
 - *Create the Tokens*
 - *Upload your Tokens to the PiCaS server*
 - *Run the example locally*

- [Run the example on the Grid](#)
- [Checking failed jobs](#)

4.4.1 Problem description

More about Picas in practice?

See also:

Check out our mooc videos Picas examples [Part I](#) and [Part II](#).

In this example we will implement the following pilot job workflow:

- First we define and generate the application tokens with all the necessary parameters.
- Then we define and create a shell script to process one task (*process_task.sh*) that will be sent with the job using the input sandbox. This contains some boiler plate code to e.g. setup the environment, download software or data from the Grid storage, run the application etc. This doesn't have to be a shell script, however, setting up environment variables is easiest when using a shell script, and this way setup scripts are separated from the application code.
- We also define and create a Python script to handle all the communication with the token pool server, call the *process_task.sh* script, catch errors and do the reporting.
- Finally we define the JDL on the User Interface machine to specify some general properties of our jobs. This is required to submit a batch of pilot jobs to the Grid that will in turn initiate the Python script as defined in the previous step.

Prerequisites

To be able to run the example you must have:

- All the three Grid *Prerequisites* (User Interface machine, Grid certificate, VO membership)
- An account on PiCaS server (send your request to [<helpdesk@surfsara.nl>](mailto:helpdesk@surfsara.nl))

4.4.2 Picas sample example

- Log in to the UI and download the *pilot_picas_fractals.tgz* example, the couchdb package for Python *couchdb.tgz* and the fractals source code *fractals.c*.
- Untar *pilot_picas_fractals.tgz* and inspect the content:

```
$tar -xvf pilot_picas_fractals.tgz
$cd pilot_picas_fractals/
$ls -l
-rwxrwxr-x 1 homer homer 1247 Jan 28 15:40 createTokens
-rw-rw-r-- 1 homer homer 1202 Jan 28 15:40 createTokens.py
-rw-rw-r-- 1 homer homer 2827 Jan 28 15:40 createViews.py
-rw-rw-r-- 1 homer homer 462 Jan 28 15:40 fractals.jdl
drwxrwxr-x 2 homer homer 116 Jan 28 15:40 sandbox
```

Detailed information regarding the operations performed in each of the scripts below is embedded to the comments inside each of the scripts individually.

- Also download the current PiCaS version `picas.tar` and put both PiCaS and the `couchdb.tgz` file in the `sandbox` directory:

```
$cd sandbox
$mv ../../couchdb.tgz ./
$mv ../../picas.tgz ./
```

- And finally compile the `fractals` program (and put it in the `sandbox` directory) and move one directory up again:

```
$cc ../../fractals.c -o fractals -lm
$cd ..
```

The `sandbox` directory now holds everything we need to send to the Grid worker nodes.

Prepare your Tokens

Create the Tokens

This example includes a bash script (`./createTokens`) that generates a sensible parameter file, with each line representing a set of parameters that the `fractals` program can be called with. Without arguments it creates a fairly sensible set of 24 lines of parameters. You can generate different sets of parameters by calling the program with a combination of `-q`, `-d` and `-m` arguments, but at the moment no documentation exists on these. We recommend not to use them for the moment.

- After you ran the `createTokens` script you'll see output similar to the following:

```
$/createTokens
/tmp/tmp.fZ33Kd8wXK
$cat /tmp/tmp.fZ33Kd8wXK
```

Upload your Tokens to the PiCaS server

Now we will start using PiCaS. For this we need the downloaded CouchDB and PiCaS packages for Python and set the hostname, database name and our credentials for the CouchDB server:

- Edit `sandbox/picasconfig.py` and set the PiCaS host URL, database name, username and password.
- Link the `picasconfig.py` file in the current directory. This makes it available for the scripts that need to upload the tokens to CouchDB:

```
$ln sandbox/picasconfig.py
```

- Make the CouchDB package locally available:

```
$tar -xvf sandbox/couchdb.tgz
```

- Upload the tokens:

```
$python createTokens.py /tmp/tmp.fZ33Kd8wXK
```

- Check your database in this link:

https://picas.surfsara.nl:6984/_utils/database.html?homerdb

replace homerdb with your Picas database name

- Create the Views (pools) - independent to the tokens (should be created only once):

```
$python createViews.py
```

Run the example locally

- If you submit the jobs on the UI, the job will start fetching tokens from the pool server and run the application locally on the UI machine:

```
$cd sandbox/
$./startpilot.sh

Connected to the database homerdb sucessfully. Now starting work...
-----
Working on token: token_2
lock 1453570581
_rev 2-8d7f141114b7335b50612ba4dfb92b3d
hostname ui
exit_code
scrub_count 0
done 0
input -q 0.100 -d 256 -m 8400
output
_id token_2
type token
-----
/usr/bin/time -v ./process_task.sh "-q 0.100 -d 256 -m 8400" token_2 2> logs_token_2.err_
↪1> logs_token_2.out
-----
Working on token: token_6
lock 1453570589
...
```

You can monitor the progress for the Tokens that are waiting, running, finished or in error state, from the PiCaS website here:

https://picas.grid.sara.nl:6984/_utils/

While the UI has started processing tokens, submit the pilot jobs to the Grid. Continue to the next section ...

Run the example on the Grid

- Create a proxy:

```
$dirac-proxy-init -b 2048 -g lsgrid_user -M lsgrid --valid 168:00 # replace lsgrid with_
↪your VO
```

- Submit the pilot jobs:

```
$dirac-wms-job-submit fractals.jdl -f jobIDs
```

It will recursively generate an image based on parameters received from PiCas. At this point, some of your tokens are processed on the Grid worker nodes and some of the tokens are already processed on the UI. Note that the UI is not meant for production runs, but only for testing few runs before submitting the pilot jobs to the Grid.

- Convert the UI output file to .png format and display the picture:

```
$convert output_token_6 output_token_6.png # replace with your output filename
```

For the tokens that are processed on Grid, you can send the output to the [Grid Storage](#) or some other remote location.

Checking failed jobs

While your pilot jobs process tasks, you can keep track of their progress through the CouchDB web interface. There are views installed to see:

- all the tasks that still need to be done (Monitor/todo)
- the tasks that are locked (Monitor/locked)
- tasks that encountered errors (Monitor/error)
- tasks that are finished (Monitor/done)

When all your pilot jobs are finished, ideally, you'd want all tasks to be 'done'. However, often you will find that not all jobs finished successfully and some are still in a 'locked' or 'error' state. If this happens, you should investigate what went wrong with these jobs. Incidentally, this will be due to errors with the Grid middleware, network or storage. In those cases, you can remove the locks and submitting some new pilot jobs to try again. In other cases, there could be errors with your task: maybe you've sent the wrong parameters or forgot to download all necessary input files. Reviewing these failed tasks gives you the possibility to correct them and improve your submission scripts. After that, you could run those tasks again, either by removing their locks or by creating new tokens if needed and then submitting new pilot jobs.

4.5 Pilot jobs

In this page we will show you how to run pilot jobs on the Grid and track the status of hundreds jobs running at a time:

Contents

- *Pilot jobs*
 - *About pilot jobs*
 - *Pilot Job Workflow*
 - * *Pilot job database*
 - *Pilot job advantages*
 - *Pilot job submission*
 - *Pilot Job Frameworks*

4.5.1 About pilot jobs

More about Pilot jobs?

See also:

Check out our mooc video [Pilot Job Frameworks](#)

When you have tens, hundreds of jobs that you submit to the Grid you may find yourself writing software which checks the status of all those jobs and tries to find out which ones have succeeded and which ones should be resubmitted.

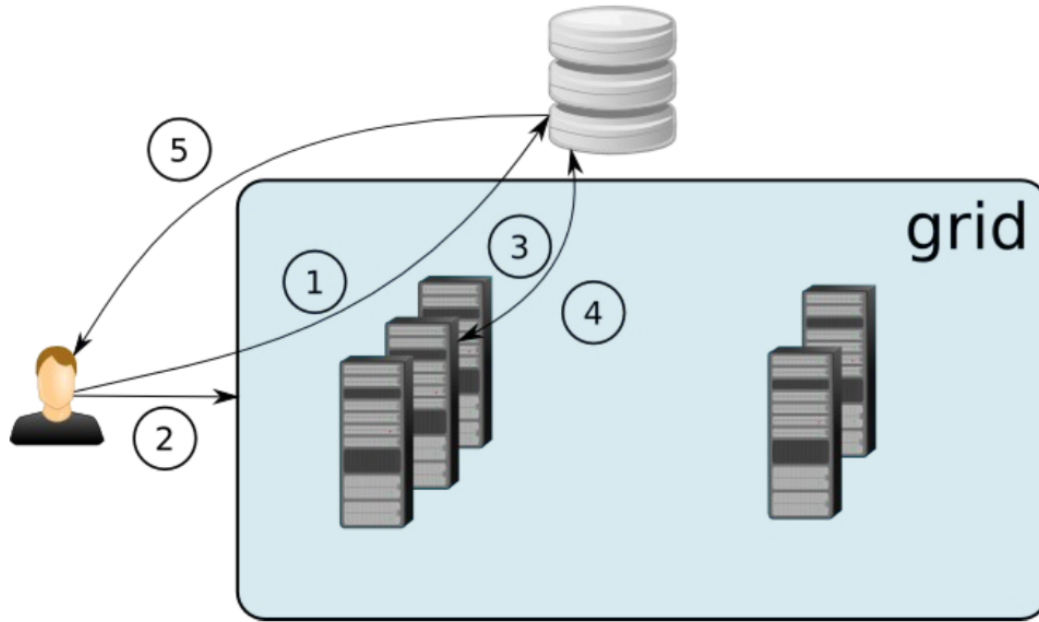
In addition, when submitting a large number of jobs you will often find that they need to operate on the same database or some other static datasource. For every jobsubmission this database then has to be copied to the node the job is running on. This introduces a large overhead per job.

A solution to these problems is to use a pilot job framework. Such frameworks start by submitting a number of pilot jobs to the Grid. Pilot jobs are like normal jobs, but instead of executing the task directly they contact a central server once they are running on a worker node. Then, and only then, will they be assigned a task, get their data and start executing. The central server handles the request from pilot jobs and keeps a log of what tasks are being handled, are finished, and can still be handed out. The pilot job can request a new task, report the successful completion of a task just executed or report that it is still working on the task it received previously. When the task server doesn't hear about the progress or completion of a task it has distributed, it will assume the pilot job is either dead or the job has failed. As a result the task will be assigned to another pilot job after it has made a request for a new task.

A pilot job will not die after it has successfully completed a task, but immediately ask for another one. It will keep asking for new jobs, until there is nothing else to do, or its wall clock time is up. This reduces overhead for jobsubmission considerably.

4.5.2 Pilot Job Workflow

The picture below illustrates the workflow of pilot job systems: (1), the user uploads work to the central database. This could be a list of parameter settings that need to be executed using some application on a computational resource. The user then (2) submits jobs just containing the application to the Grid, which handles retrieving from (3) and updating of (4) the job database. This process continues until all the work present in the database has been done, the application has crashed or the job has run out of time on the computational resource. When all work has been done, the user retrieves (5) the results from the database.



Pilot job database

In *pilot frameworks*, the database server is a server which is not necessarily tied to Grid infrastructure. In fact, you can have simple access to a pilot job server with your Internet browser or with HTTP command line clients like `wget` and `curl`.

The concept of token pools can also be used to create workflows. A task server can deal out tokens from different pools in succession and pilot jobs can remove tokens from pools (by processing them) and create tokens in other pools.

4.5.3 Pilot job advantages

Notice that when you use *pilot frameworks* you do not have to worry about job failures. When jobs fail they will not request and process new tasks. The user should not look at the pilot jobs he/she has submitted but to the number of tasks being processed. There is no need to keep track of submitted (pilot) jobs and resubmit those (and only those) that failed. Remember a pilot job will die if all tasks have been processed.

A second advantage of using pilot jobs is the reduced overhead of job submission. Once a pilot job is in place it will process tasks as long as there are any, or it's wall clock time (the maximum time a job is allowed to run) is up. This can also be advantageous for jobs who all need to transfer a large datafile. In the last example, a database should be downloaded from a SE to the Worker Node where the job is running. Note, that this is always the same database for all jobs. When a pilot job is running and is processing task after task it only needs to download the database once.

4.5.4 Pilot job submission

To be able to submit multiple pilot jobs at once, they are submitted to the Grid with a specific *JDL* type called Parametric. Learn more about this technique in parametric jobs section.

4.5.5 Pilot Job Frameworks

There are several pilot frameworks for the Grid. At SURFsara we support PiCaS:

- *Picas Overview*
- *Picas Example*

4.6 GPU jobs

In this page we will show you how to run jobs that use GPUs on the Grid:

Contents

- *GPU jobs*
 - *About GPU jobs*
 - *GPU job submission*
 - *Quick GPU example*
 - *Example GPU Job*

4.6.1 About GPU jobs

Certain problems you want to run on special hardware to decrease the runtime of your program, such as on a GPU. The Grid has started to support and run GPUs jobs. This section contains the best practices of running GPU jobs on the Dutch Grid, as supported by SURF.

4.6.2 GPU job submission

To submit a GPU job to the Grid, the *JDL* file must contain a tag field set to `gpu`. This field looks like the following:

```
Tags = {"gpu"};
```

This will put your job in the GPU queue after which the job lands on a compute element (CE) that contains a GPU and can run the relevant code.

4.6.3 Quick GPU example

To quickly run the example, download the shell-script and jdl-file as given below. For the full explanation on what the job just did, see the instructions in the [Example GPU Job](#) section.

- Copy the shell-script `gpu_job.sh` to your UI directory:

```
$wget http://doc.grid.surfsara.nl/en/latest/_downloads/gpu_job.sh
```

- Copy the jdl-file `gpu_job.jdl` to your UI directory:

```
$wget http://doc.grid.surfsara.nl/en/latest/_downloads/gpu_job.jdl
```

And submit the job with:

```
$dirac-wms-job-submit gpu_job.jdl
JobID = 123
```

And inspect the output after retrieving the files with:

```
$dirac-wms-job-get-output 123
```

4.6.4 Example GPU Job

An example of a GPU job starts with the code we want to run. For this example, we decide to run a CUDA example made by Nvidia, which can be found at [github](#). To ensure the code runs on the Grid, it is containerized with `apptainer` and the container is distributed through [CVMFS](#).

To build the container yourself, you can run

```
$apptainer build --fakeroot --nv --sandbox cuda_example_unpacked.sif cuda_
↪example.def
```

on a machine with `apptainer` installed, where the user has `fakeroot` privileges. The `cuda_example.def` definitions file contains the following recipe:

```
Bootstrap: docker
From: nvidia/cuda:11.8.0-devel-centos7

%post
#This section is run inside the container
yum -y install git make
mkdir /test_repo
cd /test_repo
git clone https://github.com/NVIDIA/cuda-samples.git
cd /test_repo/cuda-samples/Samples/2_Concepts_and_Techniques/eigenvalues/
make

%runscript
#Executes when the "apptainer run" command is used
#Useful when you want the container to run as an executable
cd /test_repo/cuda-samples/Samples/2_Concepts_and_Techniques/eigenvalues/
./eigenvalues
```

(continues on next page)

(continued from previous page)

```
%help
This is a demo container to show how to build and run a CUDA application
on a GPU node
```

This will create a container with the compiled eigenvalues example inside, which makes use of an Nvidia GPU to calculate the eigenvalues of a 2048 x 2048 matrix.

This container has already been distributed on CVMFS and can be found at `/cvmfs/softdrive.nl/lodewijkn/cuda_example_unpacked.sif`.

To run this container on the Grid, making use of the GPUs, the following jdl has to be submitted to DIRAC, or the workload management system (WMS) of your choice. This jdl runs a shellscript on the CE, calling the container on CVMFS. This script, called `gpu_job.sh` is given as:

```
#!/bin/bash

cat /proc/self/status | grep 'Cpus_allowed_list:'
echo
/usr/bin/nvidia-smi

pwd
hostname

/cvmfs/oasis.opensciencegrid.org/mis/apptainer/bin/apptainer run --nv /cvmfs/
↪softdrive.nl/lodewijkn/cuda_example_unpacked.sif
```

In the command that calls apptainer the `--nv` flag is necessary to expose the GPU to the container. The `gpu_job.sh` script is then passed along inwith the jdl to the Grid when submitting the job. The jdl, called `gpu_job.jdl`, is given next:

```
[
  JobName = "my_gpu_job";
  Executable = "gpu_job.sh";
  StdOutput = "StdOut";
  StdError = "StdErr";
  InputSandbox = {"gpu_job.sh"};
  OutputSandbox = {"StdOut", "StdErr"};
  Site = "GRID.SURF.nl";
  Tags = {"gpu"};
  CPUTime = 600;
]
```

To submit the job, ensure all the necessary files are available: the `gpu_job.sh` script and `gpu_job.jdl` jdl file. Then submit the job with:

```
$dirac-wms-job-submit gpu_job.jdl
```

After the job has run succesfully, the stdout output looks like:

```
Cpus_allowed_list: 11-21

Thu Dec  8 14:57:12 2022
+-----+
| NVIDIA-SMI 515.65.01      Driver Version: 515.65.01    CUDA Version: 11.7     |
```

(continues on next page)

(continued from previous page)

-----+-----+-----									
GPU Name		Persistence-M		Bus-Id		Disp.A		Volatile Uncorr. ECC	
Fan Temp Perf		Pwr:Usage/Cap		Memory-Usage		GPU-Util		Compute M.	
								MIG M.	
=====									
0 NVIDIA A10		Off		000000000:00:07.0 Off				0	
0% 42C P0		57W / 150W		0MiB / 23028MiB		3%		Default	
								N/A	
-----+-----+-----									
-----+-----+-----									
Processes:									
GPU		GI CI		PID Type		Process name		GPU Memory	
		ID ID						Usage	
=====									
No running processes found									
-----+-----+-----									
/tmp/ZTiMDmYBVN2nCifV3nVLvLKmABFKDmABFKDmURGKDmABFKDm5i8MKo/DIRAC_hA3xkNpilot/									
↪20									
wn-a10-04.gina.surfsara.nl									
Starting eigenvalues									
GPU Device 0: "Ampere" with compute capability 8.6									
Matrix size: 2048 x 2048									
Precision: 0.000010									
Iterations to be timed: 100									
Result filename: 'eigenvalues.dat'									
Gerschgorin interval: -2.894310 / 2.923303									
Average time step 1: 0.987820 ms									
Average time step 2, one intervals: 1.169709 ms									
Average time step 2, mult intervals: 2.616700 ms									
Average time TOTAL: 4.785920 ms									
Test Succeeded!									

And you have run your first GPU job on the Grid!

SERVICE IMPLEMENTATION

5.1 Downtimes and maintenances

Here you will find details to keep informed for the ongoing and upcoming downtimes and maintenances in our Grid systems:

Contents

- *Downtimes and maintenances*
 - *Ongoing maintenances*
 - *Upcoming maintenances*

5.1.1 Ongoing maintenances

You can lookup announcements for the current status of the Grid systems, including all ongoing downtimes and maintenances on [our website](#) under:

Systems > System status > National e-Infrastructure Grid Maintenances

Direct link: <https://ganglia.grid.surfsara.nl/cgi-bin/eInfra.py>

5.1.2 Upcoming maintenances

If you use want to receive notifications and be prepared for upcoming downtimes and maintenances, you can create personal subscriptions to receive announcements tailored specifically for the Grid services and clusters that are relevant for you.

- Login to the portal here: <https://operations-portal.egi.eu/> (click on EGI button)
- Select your academic or social account and login
- If your academic or social account does not work go to 'EGI SSO' for and select 'forgot password' to create a new account
- **The first time you login you will be asked to “Sign Up EGI user community”**
 - Click and review the terms & conditions. Once you agree you will receive an email. Click on the link to verify address
 - Accept the invitation and verify again

- You should have now logged in successfully to the portal
- Click on Downtimes Link button and then on the top right “SUBSCRIPTION”. Add your email as Subscriber
- **Create the following rule to receive notifications**
 - Rule: I WANT
 - Region: NGI_NL
 - Site: SARA-MATRIX
 - Node: All nodes
 - VO: All VOs
 - Comm. channels: click on ‘Add comm’. Then ‘+Add new’ -> Type: Email(HTML), Value: put your email address and click ‘Enter’ in the same field box -> click ‘Close’
- Click on *Save rules specifications* to take effect. From now on you should receive notifications when the Gina cluster or SURFsara dCache has a downtime

NOTE: for dCache only select webdav and srm only notifications.

If you need help creating your subscription(s) we are of course willing to assist at helpdesk@surfsara.nl.

5.2 Statistics and monitoring

In this section you will find information about the Grid systems status and usage statistics:

Contents

- *Statistics and monitoring*
 - *SURFsara systems*
 - *NIKHEF facility*
 - *EGI accounting and operations portals*

5.2.1 SURFsara systems

The webpage below contains monitoring information about the usage of various systems at SURFsara:

- [SURFsara systems usage](#)

More specific monitoring information about the Grid jobs usage can be found in the following website:

- [SURFsara Grid systems](#)

5.2.2 NIKHEF facility

The webpage below contains Nikhef facility statistics and status overview:

- [Statistics NIKHEF facility](#)

5.2.3 EGI accounting and operations portals

The portal below provides information about accounting statistics for sites, regions, VOs:

- [EGI Accounting portal](#)

The following site provides Grid sites info updated by participating NGIs:

- [GOCDB NGIs info](#)

TUTORIALS

6.1 MOOC - Introduction to Grid Computing

This page includes the complete course material as presented during the SURFsara MOOC Introduction to Grid Computing:

Contents

- *MOOC - Introduction to Grid Computing*
 - *About*
 - *Lectures*
 - *Use cases*
 - *Animations*
 - *EGI-InSPIRE*

6.1.1 About

As of 18 November 2013, SURFsara offered the Massive Open Online Course (MOOC) Introduction to Grid Computing. The course aimed to help researchers to understand the Grid key concepts and the role of Grid Computing in computationally intensive problems. This included working with portals, workflow management systems and the Grid middleware.

The taught material consists of:

- a set of **video lectures**.
- a set of **quizzes, assignments and exercises**.
- real world examples of **use cases** on Grid.

Note: The mooc participants were provided with student accounts and a preconfigured Virtual Machine (VM) with all the necessary Grid tools installed. If you want to run the examples presented in the video lectures below you will have to request your personal Grid account, see *Prerequisites*. Please contact helpdesk@surfsara.nl if you need help with this.

6.1.2 Lectures

The entire course video lectures can be found in the [MOOC Video lectures](#).

Course Overview

Course Overview slides pdf

Introduction to Parallel and Distributed Computing

Intro to Parallel and Distributed Computing pdf

Cluster Computing

Cluster Computing pdf

Grid Computing Overview

Grid Computing Overview pdf

Grid_Glossary

Grid Glossary pdf

Hands-on set

Exercises Distributed Cluster Grid pdf

Quiz Distributed Cluster Grid pdf

Working Environment - Grid prerequisites

Working Environment_I Grid prerequisites pdf

Working Environment - Remote access

Working Environment II Remote access pdf

Code gridpi tar

Grid Certificate - Security

Grid Certificate I (security) pdf

Grid Certificate I (extras) pdf

Obtain a Grid Certificate

Sectigo: new way to request a certificate!

Note: Sectigo allows you to get your Grid certificate instantly from the GEANT Trusted Certificate Service (instead of Terena portal), by using your institutional login and SURFconext. Read the [User Guide](#) or login directly on the [Sectigo portal](#).

Grid Certificate II (Obtaining a certificate) pdf

User Interface machine

User Interface machine pdf

Virtual Organisations

Virtual Organisations pdf

Hands-on set

Exercises Install Certificate pdf

Quiz_Install Certificate pdf

Grid job Lifecycle

Grid job Lifecycle pdf

Start a Grid_Session

Start a Grid Session pdf

My First Grid job

My First Grid job pdf

Code MyFirstJob tar

Grid Toolkit

Grid Toolkit pdf

Hands-on set

Quick start guide pdf

Exercises First Grid job pdf

Quiz_First Grid job pdf

Application_submission to Grid I

Application submission to Grid I script pdf

Code script tar

Application_submission to Grid II

Application submission to Grid II executable pdf

Code compiled tar

Advanced Grid jobs I

Advanced Grid jobs I Collections & Parametric pdf

Code Collections Parametric tar

Advanced Grid jobs II

Advanced Grid jobs II Multicore pdf

Code multicore tar

Data parallel processing Hadoop

Data parallel processing Hadoop pdf

Hands-on set

Exercises Advanced Jobs pdf

Quiz Advanced Jobs pdf

lcg/lfc/lfn? Only for large files with multiple replicas.

The lectures Data Management on the Grid [1-3] present the lcg/lfc/lfn *Storage clients*. However, we advise you to better use the *globus client* or *srm client* tools, unless you need to run jobs on multiple sites which require access on the **same** large dataset (or database). In case of doubts, contact us at helpdesk@surfsara.nl.

Data Management on the Grid I

Data Management on the Grid I pdf

Data Management on the Grid II

Data Management on the Grid II pdf

Data Management on the Grid III

Data Management on the Grid III pdf

Code DMlargefiles tar

Storage Resource Manager

Storage Resource manager pdf

Code DMsrm tar

Hands-on set

Exercises_Data Management pdf

Quiz_Data Management pdf

Introduction to Workflows I

Introduction to Workflows I pdf

Introduction to Workflows II

Introduction to Workflows II pdf

WS-Pgrade I

WSpgrade I pdf

WS-Pgrade II

WSpgrade II pdf

Science Gateways

Science Gateways pdf

Hands-on set

Exercises Workflows pdf

Code Exercises Workflows tar

Code Solutions Workflows tar

Quiz Workflows pdf

Pilot Job Frameworks

Pilot job frameworks pdf

Picas Server side I

Picas server side I pdf

Picas Server side II

Picas server side II pdf

Picas client side

Picas client side pdf

Picas practise I

Code Picas tar

Picas practise II

Code Picas tar

Course Summary

Course summary pdf

6.1.3 Use cases

Use case: EGI overview

EGI overview pdf

Use case: LOFAR

Extreme physics in space pdf

Use case: Climate change over Europe

Climate change pdf

Use case: VisIVO Science Gateway

VisIVO Science Gateway pdf

Use case: Picas

Picas pdf

Use case: Molecular Biology

Molecular Biology pdf

6.1.4 Animations

We have prepared a set of animations to display the basic usage of Grid Infrastructure. The animations were presented during the lectures.

6.1.5 EGI-InSPIRE

The work is supported by the EGI-InSPIRE project (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe), co-funded by the European Commission (contract number: RI-261323) for four years from the 1st of May 2010. EGI-InSPIRE is a collaborative effort involving more than 50 institutions in over 40 countries. Its mission is to establish a sustainable European Grid Infrastructure (EGI).

- *MOOC - Introduction to Grid Computing*
- *gLite tutorial 2008*

FREQUENTLY ASKED QUESTIONS

7.1 Frequently asked questions

Check out in this page the most commonly asked questions about Grid. If you still have questions, please contact us at helpdesk@surfsara.nl:

Contents

- *Frequently asked questions*
 - *Getting started*
 - * *I never worked with the Grid before. Where is a good starting point?*
 - * *Where can I lookup up Grid terms?*
 - *Certificates*
 - * *How can I change my Grid certificate password?*
 - * *Unable to load certificate error*
 - * *What are the correct permissions for my certificate files?*
 - * *Couldn't find valid credentials error*
 - * *Get non-vomsified proxy locally*
 - * *How can I renew my certificate?*
 - * *Does my key match the certificate?*
 - * *What is the expiry date of my certificate?*
 - * *What is the expiry date of my VO membership?*
 - * *How can I see the subject of my certificate?*
 - *Using resources*
 - * *How many cpu's, nodes does the Grid offer?*
 - * *How many cpu hours are available?*
 - * *What is the average amount of memory available per node?*
 - * *What is the data transfer speed between Grid locations?*
 - * *How can I calculate the total CPU time I consumed?*

- * *System usage and CPU efficiency*
- * *How can I find all the available Storage Elements and get their SURLS?*
- * *How can I find all the available Compute Elements and use in my JDL?*
- * *How to run PBS jobs with wallclock greater than 36 hours on local clusters?*
- * *How to use the Grid worker node /scratch on Gina?*

7.1.1 Getting started

I never worked with the Grid before. Where is a good starting point?

New users are advised to read through our tutorial page, the *Prerequisites* and first-grid-job which guides you through the whole process from getting a Grid certificate to the actual job run. The Grid team at SURFsara is willing to assist, just contact helpdesk@surfsara.nl.

Where can I lookup up Grid terms?

Check out the file Grid Glossary pdf that contains most of the basic Grid terminology and abbreviations.

7.1.2 Certificates

How can I change my Grid certificate password?

Before you create a new private key file with a new password, we recommend you to make a backup of the old userkey.pem file.

To change your Grid certificate password, type:

```
$openssl rsa -in ~/.globus/userkey.pem -des3 -out ~/.globus/new_private_key_file
$mv ~/.globus/new_private_key_file ~/.globus/userkey.pem # this will replace your old
↪key file with the old password!
```

Note: this only changes the password you use for your certificate. If you think your certificate is compromised, you HAVE to revoke your certificate!

Unable to load certificate error

If you get the following error:

```
unable to load certificate 17714:error:0906D064:PEM routines:PEM_read_bio:bad base64
decode:pem_lib.c:781:
```

when you use the command `openssl x509 -text -noout -in usercert.pem`, it means that the email with the certificate wasn't saved properly as plain text (it included the Mime type for formatting). Repeat carefully the steps as described in *Retrieve your DutchGrid certificate* section.

What are the correct permissions for my certificate files?

- Set the proper permissions to your certificate files:

```
$chmod 644 usercert.pem
$chmod 400 userkey.pem
```

- Verify the correct permissions:

```
$cd $HOME/.globus
$ls -l

-rw-r--r--    1 homer    homer          4499  May 10 13:47  usercert.pem
-r-----    1 homer    homer           963  May 10 13:43  userkey.pem
```

Note that the private key file should be **read-only** and only readable to you.

Couldn't find valid credentials error

If you get the following error when creating a new proxy:

```
ERROR: Couldn't find valid credentials to generate a proxy.
Use --debug for further information.
```

The permissions on your installed certificate are probably wrong. Set the *correct permissions* and try creating a proxy again.

Get non-vomsified proxy locally

- To download locally the proxy stored on MyProxy server you need to set a passphrase upon creation. To do this, protect your proxy with a MyProxy pass phrase by omitting option “-n”:

```
$myproxy-init -d
```

It will first ask your Grid certificate password and then prompt you to enter a MyProxy passphrase twice. You will use the latter passphrase to download your proxy.

Here is an example of the displayed output:

```
Your identity: /O=dutchgrid/O=users/O=sara/CN=Homer Simpson
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Proxy Verify OK
Your proxy is valid until: Wed Jan 13 14:35:00 2016
Enter MyProxy pass phrase:
Verifying - Enter MyProxy pass phrase:
A proxy valid for 168 hours (7.0 days) for user /O=dutchgrid/O=users/O=sara/
↪CN=Homer Simpson now exists on px.grid.sara.nl.
```

- Now use the MyProxy pass phrase to get this proxy locally on the UI:

```
$myproxy-get-delegation -d
```

Here is an example of the displayed output:

```
Enter MyProxy pass phrase:
A credential has been received for user /O=dutchgrid/O=users/O=sara/CN=Homer_
↪Simpson in /tmp/x509up_u39111.
```

Note that the downloaded proxy will not include the voms attributes.

How can I renew my certificate?

The personal Grid certificates are valid for a year. This means that every year you need to renew your personal Grid certificate. The procedure for renewing your certificate depends on your CA, either Sectigo or DutchGrid.

- For *Sectigo* Grid certificate, you can request a new certificate anytime from the [Sectigo portal](#). Follow this guide to *obtain and install a Sectigo Grid certificate*.
- For *DutchGrid* Grid certificate, you have two options:
 - When your certificate has already expired, you *have* to request a new certificate from scratch with the jGridstart tool. Follow this guide to *obtain a DutchGrid certificate*.
 - If your current certificate has *not* expired yet, you can *renew* your certificate. This is a faster procedure because you avoid revisiting your RA for your id verification. What you need to do:
 1. Log in to the UI with X session enabled.
 2. Start the jGridstart tool on the UI (assuming that your current certificate is installed there): `java -jar jgridstart-wrapper-XX.jar`
 3. Select **Actions** -> **Renew** from the menu bar.
 4. Generate a new request by verifying your details (name, surname, email, organisation). At this stage you will provide a new password for your new Grid certificate - make sure you keep this safe! Click “Next”.
 5. Submit the request. This will create a new private `userkey.pem` file in your `~/.globus` directory. Click “Next”.
 6. You will receive your new certificate within few days via email. Once received, follow the instructions to *install it on the UI*.

Keep in mind that when you renew your certificate the certificate key will change too. To avoid mixing up the old and new certificate files, check whether your new certificate and key *match each other*.

Does my key match the certificate?

Using the modulus you can see whether a key and a certificate match. The modulus is a short message which can be used to identify a private key and the key which was signed with the certificate. If they match, the certificate signs that private key. If not, you may have mixed up different key or certificate files.

To find the modulus of your key, use:

```
$openssl rsa -in userkey.pem -noout -modulus
```

which requires the key which you used to protect your key file. To find the modulus of your certificate, use:

```
$openssl x509 -in usercert.pem -noout -modulus
```

If the moduli of the key file and the certificate file do not match, you cannot use that combination to identify yourself.

What is the expiry date of my certificate?

To find out when your certificate is valid, use:

```
$openssl x509 -in usercert.pem -noout -dates
```

This will tell you when your certificate is valid.

Note that a key does not have a validity period.

What is the expiry date of my VO membership?

Your VO membership needs to be renewed every year, which includes signing of the (possibly updated) Acceptable Use Policy for the VO.

Please take care to renew your membership within a month after expiry. This prevents you from having to rejoin the VO, as all expired memberships will be automatically removed by the system after a month.

How can I see the subject of my certificate?

The subject of a certificate is the human-readable identification of who the certificate belongs to. It usually contains your name, country, organisation and your e-mail address.

To find out who the certificate belongs to, use:

```
$openssl x509 -in usercert.pem -noout -subject
```

7.1.3 Using resources

How many cpu's, nodes does the Grid offer?

The Grid infrastructure is interconnected clusters in Netherlands and abroad. The users can get access to multiple of these clusters based on their *Virtual Organisation*.

- Global picture: 170 datacenters in 36 countries: in total more than 330000 compute cores, 500 PB disk, 500 PB tape.
- In the Netherlands NGL_NL infrastructure: 14 datacenters (3 large Grid clusters, 11 smaller ones): in total approximately 10000 compute cores, 12 PB disk, tape capacity up to 170 PB.

How many cpu hours are available?

The available core hours and storage depend on the funding models. We make tailored agreements to incorporate the user requirements and grant resources based on the applicable funding scheme.

What is the average amount of memory available per node?

The average memory per node depends on number of cores per node. It is typically 8GB per core, but the nodes vary between 12 and 64 cores per node (48 to 256GB RAM per node).

What is the data transfer speed between Grid locations?

In the Netherlands NGL_NL infrastructure the transfer speed between Grid storage and Grid processing cluster (at SURFsara) is up to 500Gbit/s. The transfer speed between nodes is 10Gbit/s and between sites it is typically 10 to 20 Gbit/s.

How can I calculate the total CPU time I consumed?

The total CPU time depends on the amount of cores that your application is using and the wallclock time that the corresponding job takes to finish:

```
CPU time = #cores x wallclock(per job) x #jobs
```

For example, let's say that a single job takes 12 h to finish on a 4-core machine and we submitted 10,000 of those. The total CPU time spent is:

```
CPU time = 4cores x 12h x 10,000 = 480,000 CPU hours ~ 55 CPU years
```

System usage and CPU efficiency

CPU efficiency is an important factor to detect if the jobs run smoothly on the infrastructure. The CPU efficiency depends on the real CPU usage and the WallClock time for the job to finish:

```
CPU efficiency = CPU time / WallClock time
```

If the CPU was efficiently being used during the job runtime, then a single core job will have efficiency close to 100%. For multicore jobs the efficiency is higher than 100%.

How can I find all the available Storage Elements and get their SURLS?

- To find out the available SEs for a certain VO, type:

```
$lcg-infosites --vo lsgrid se
```

How can I find all the available Compute Elements and use in my JDL?

- To find out the available CEs (Compute Elements) for a certain VO, type:

```
$lcg-infosites --vo lsgrid ce
```

Note here that the Total, Running and Waiting numbers are per queue, and the CPU and Free number are per cluster.

- To specify a specific cluster in your JDL file, use the following syntax:


```
Requirements = (RegExp("rug",other.GlueCEUniqueID)); # this requires the job to
↳ land on the "rug" site

# or you can specify the full UI hostname
Requirements = RegExp("gb-ce-lumc.lumc.nl",other.GlueCEUniqueID); # job lands at
↳ lumc
```

How to run PBS jobs with wallclock greater than 36 hours on local clusters?

In order to run PBS (Portable Batch System) jobs that last more than 36 hours, you need to select the proper queue with the `-q` flag in your `qsub` command when submitting the job:

- If you do *not* use `-q` flag and `lwalltime` directive, then the medium queue is picked and jobs lasting more than 36 hours will be killed.
- If you do *not* use `-q` flag but specify `-lwalltime` directive with value larger than 36 hours, then you request more walltime than the max walltime available in the default medium queue and the job does not start at all.
- If you use the `-q` flag, it is sufficient to get your jobs running for the amount of hours that the specified queue permits.

How to use the Grid worker node /scratch on Gina?

You should not write data directly under the worker node `/scratch`, but use your job directory instead. See the instructions here to make efficient use of the local storage on the Grid worker nodes.

7.2 Troubleshooting

Contents

- *Troubleshooting*
 - *General troubleshooting steps*
 - *How can I get more logging info for my job?*
 - *File transfers are stuck*

7.2.1 General troubleshooting steps

Don't hesitate to contact us when things go wrong! We're happy to help you overcome the difficulties that every Grid user faces.

In order to assist you better, we have a few troubleshooting steps that may already get you going and otherwise may help us to help you.

- Check the output of `voms-proxy-info -all`. Is your proxy still valid? Does it have the correct attributes for the work you're doing?
- Try running your command with higher debugging level or verbosity.

```
$glite-wms-job-submit --debug ...
$rmcp -debug ...
$gfal-copy --verbose ...
$globus-url-copy -debugftp -verbose-perf -verbose ...
$curl --verbose ...
```

- Is the resource you're using in *downtime*?
- Can you connect to the service?

```
## A basic firewall check: can you connect to the port?
$telnet srm.grid.sara.nl 8443

## Testing the SSL layer of a connection to the dCache SRM door
$echo 'QUIT' | openssl s_client -connect srm.grid.sara.nl:8443 \
-CApath /etc/grid-security/certificates
## One of the last lines should be: 'Verify return code: 0 (ok)'

## Testing a gridFTP door, control channel
$telnet rabbit1.grid.sara.nl 2811

## GridFTP data channels are more difficult to test, because the port opens only
↳ after a transfer is initiated.
## But after we start an iperf service, you can try to telnet to it.
$telnet rabbit1.grid.sara.nl 24000

## Or just test with iperf:
$iperf3 -c rabbit1.grid.sara.nl -p 24000
$iperf3 -c rabbit1.grid.sara.nl -p 24000 --reverse
## Keep in mind that we have to start iperf first!
```

7.2.2 How can I get more logging info for my job?

To find out more info about the status of your job, use:

```
$glite-wms-job-logging-info -v 2 https://wms2.grid.sara.nl:9000/PHeyedC1EYBjP9l_Xq9mQ #
↳ replace with your job URL
```

And if you use a file to store your jobs, run:

```
$glite-wms-job-logging-info -v 2 -i jobIds # replace jobIds with your file
```

7.2.3 File transfers are stuck

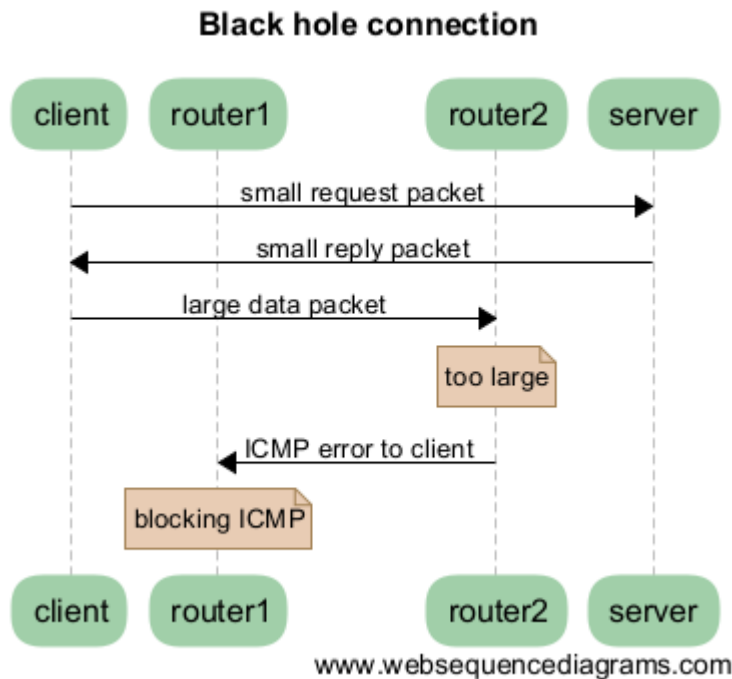
Occasionally, transfers are stuck when 0 bytes have been transferred. There are some common causes for stalled transfers.

- A firewall blocks the ports for the data channel. If you use `rmcp`, specify `--server_mode=passive`. If that doesn't help, check whether your firewall allows outgoing traffic to ports 20000 to 25000 (GridFTP data channel range).
- You've reached the maximum number of transfers for the storage pools that have been allocated to you. All transfers beyond the maximum will be queued, until previous transfers finish to make 'transfer slots' available, or

until they time out. Besides the failing transfers, there is another downside: some of your jobs might be wasting CPU time while they wait for input files. This is not efficient. It's better to reduce the number of concurrent transfers so that you don't reach the maximum, or ask us whether the maximum can be increased.

You can see whether this happens at [these graphs](#). A red color ('Movers queued') means that there are stalling transfers.

- You're transferring files from/to outside SURFsara, and your endpoint support a MTU (network packet) size of 9000, but the network path doesn't. Control traffic passes through because it consists of small packets. But data traffic consists of large packets and these are blocked. The image below illustrates this:



Some tools to test this:

```
# Run this from your endpoint of the transfer; adjust the value to find the limit.
# Check first whether your system supports a MTU of 9000.
ping -M do -s 8972 gridftp.grid.sara.nl

# This command tells you what the supported MTU value is.
tracert gridftp.grid.sara.nl
```

Another good tool for testing the network is `iperf`. We'll start an `iperf` server at your request so that you can test against it.

```
# Using iperf3 to test upload speed
iperf3 -c rabbit1.grid.sara.nl --port 24000 --parallel 4

# Same but for download speed
iperf3 -c rabbit1.grid.sara.nl --port 24000 --parallel 4 --reverse

# Using the older iperf to test upload and download speed simultaneously, with 4
↪ streams
iperf -c rabbit1.grid.sara.nl --port 24001 --parallel 4 --dualtest
```

A fix for Linux servers is to enable `tcp_mtu_probing` in `sysctl.conf`. This enables the Linux kernel to select the best MTU value for a certain network route.

7.3 Documentation how-to

Tip: Do you have ideas for corrections and improvements of our user documentation? In this page you will learn how to:

- contribute to our documentation
 - edit the docs with Sphinx language
 - build the docs on your laptop
-

Our documentation is hosted on Github and is written in [Sphinx restructured text](#). Behind the scenes we use [ReadTheDocs](#) to publish it automatically. You can contribute either directly to our [Grid Github repo](#) or send an email to our helpdesk with your remarks and we will change the documentation ourselves. Any contribution is welcome!

The rest of this page explains how to submit your changes directly through Github.

7.3.1 Contribute through GitHub

In case that you have a GitHub account and a little knowledge of git (see [GitHub's git cheat sheet](#)), you can try submitting your changes directly to our repository. Here is what you have to do:

1. [Fork our Grid Github repo](#)
2. Git pull your fork
3. [Edit with Sphinx language](#) the files with your changes
4. [Build the documentation locally](#) to preview the changes
5. Commit and push your changes back to your fork
6. Create a [pull request](#) to inform us of your changes
7. After we've reviewed and accepted your work, we will merge your commits and the documentation will be updated automatically

7.3.2 Edit with Sphinx language

When you contribute directly to our Github repo we ask you to write the changes in Sphinx language. The philosophy of Sphinx documentation is that content is stored in files that can be easily read *and* edited by humans, in a format called *restructured text*, with the file extension `.rst`. Using a simple grammar, text can be styled. The document is structured using special tags; using these tags, documentation can be split into multiple files, and you can cross-reference between files and build indexes.

Although Sphinx is quite intuitive, we have created a simple Sphinx cheatsheet to help you use the Sphinx syntax:

- [Sphinx cheatsheet](#)

7.3.3 Build the documentation locally

Because the syntax of the files is human readable, you can edit the files using your favourite text editor. Once you are done editing, you can generate documentation in various formats, such as HTML or epub. While you can edit the pages on virtually any system, it is recommended to preview your changes before publishing them.

There are different ways to generate the HTML documentation from source and review your changes:

- *Docker image*
- Sphinx local installation
- *GitHub edit/preview*

Note that you only need to use one of the options mentioned above. Using Docker is the preferred way, as this mimics the ReadTheDocs build system closest. GitHub edit/preview on the other hand is good enough for minor, textual changes, but is otherwise the least preferred option.

Below you will find information for each of the methods.

Docker image

This is the preferred option to build and test your changes. It tries to build the documentation the same way as readthedocs.org.

- Install Docker image
- Once the Docker image is ready, find the following script inside your Github fork and run it to build your documentation. Provide an output location (default: `./build`) and the Docker image name (default: `readthedocs/build`):

```
./build.sh
```

Optionally you can provide an output location (default: `./build`) and the Docker image name (default: `readthedocs/build`):

```
./build.sh /alternative/output/path/ docker_image_alternative_name
```

Example:

```
./build.sh mybuild readthedocs/build:latest
```

Note: For Mac OS X, use `./build_mac.sh` instead.

Sphinx local installation

For the Sphinx documentation setup locally you will need to:

- Install Sphinx `sphinx_install`
- To generate HTML documentation, use the command:

```
make html
```

which will generate static pages in the `build`-directory as long as you have the software Sphinx installed locally.

Github edit/preview

For small changes you can edit a page directly from your GitHub fork webview. The *preview* button does not give a fully compatible *rst* overview, but is sufficient for textual changes.

7.4 Cookiebeleid

7.4.1 Wat is een cookie?

Wij maken op deze website gebruik van cookies. Een cookie is een eenvoudig klein bestandje dat met pagina's van deze website wordt meegestuurd en door uw browser op uw harde schrijf van uw computer wordt opgeslagen. De daarin opgeslagen informatie kan bij een volgend bezoek weer naar onze servers teruggestuurd worden.

7.4.2 Google Analytics

Via onze website wordt een cookie geplaatst van het Amerikaanse bedrijf Google, als deel van de “Analytics”-dienst. Wij gebruiken deze dienst om bij te houden en rapportages te krijgen over hoe bezoekers de website gebruiken. Google kan deze informatie aan derden verschaffen indien Google hiertoe wettelijk wordt verplicht, of voor zover derden de informatie namens Google verwerken. Wij hebben hier geen invloed op. Wij hebben Google niet toegestaan de verkregen analytics informatie te gebruiken voor andere Google diensten. De informatie die Google verzamelt wordt zo veel mogelijk geanonimiseerd. Uw IP-adres wordt nadrukkelijk niet meegegeven. De informatie wordt overgebracht naar en door Google opgeslagen op servers in de Verenigde Staten. Google stelt zich te houden aan de Safe Harbor principes en is aangesloten bij het Safe Harbor-programma van het Amerikaanse Ministerie van Handel. Dit houdt in dat er sprake is van een passend beschermingsniveau voor de verwerking van eventuele persoonsgegevens.

7.4.3 In- en uitschakelen van cookies en verwijdering daarvan

Meer informatie omtrent het in- en uitschakelen en het verwijderen van cookies kan je vinden in de instructies en/of met behulp van de Help-functie van jouw browser.

7.4.4 Meer informatie over cookies?

Op de volgende websites kan je meer informatie over cookies vinden:

- Cookierecht.nl
- Consumentenbond: “Wat zijn cookies?”
- Consumentenbond: “Waarvoor dienen cookies?”
- Consumentenbond: “Cookies verwijderen”
- Consumentenbond: “Cookies uitschakelen”
- Your Online Choices: “A guide to online behavioural advertising”

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`