NORDUGRID-MANUAL-6
8/12/2010

The NorduGrid/ARC User Guide

Advanced Resource Connector (ARC) usage manual

Contents

List of Figures

Chapter 1

Preface

The NorduGrid's [1] Advanced Resource Connector (ARC) is a light-weight Grid solution, designed to support a dynamic, heterogeneous Grid facility, spanning different computing resources and user communities. It provides *middleware* to interface between user applications and distributed resources.

The NorduGrid ARC middleware is based on the Globus® API, libraries and services [2]. The architecture of the ARC middleware pursues stability and robustness of services, which implies complete decentralization and functional independence of the components. In order to support this architecture, several innovative approaches were used, such as the Grid Manager [3] and the User Interface (Section 8). Several Globus components were extended and developed further, such as the Information Model [4] and XRSL – Extended Resource Specification Language (Section 7).

ARC main components are (see Figure 1.1):

- Grid services running on the resources: the Grid Manager, the `gridftpd` and the information services.
- Indexing services for the resources and data.
- Clients making intelligent use of the distributed information and data available on the Grid.

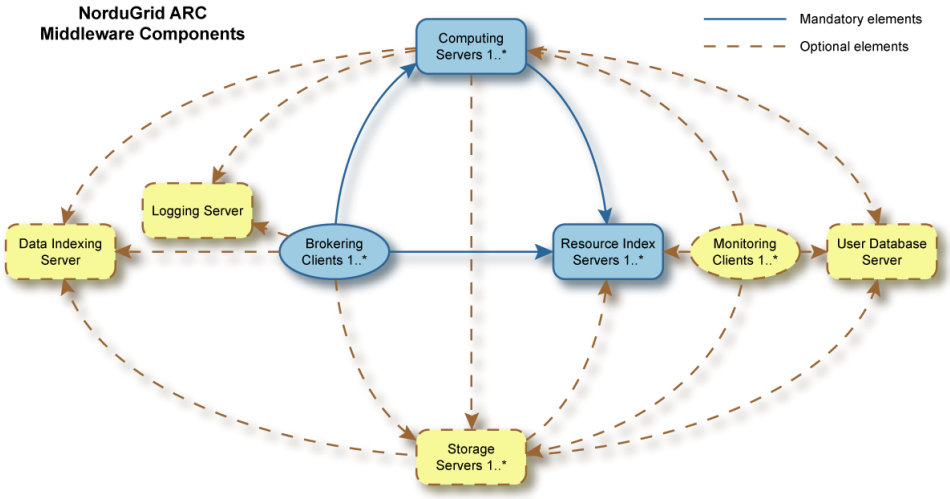

Figure 1.1: Architectural components of the ARC middleware.

This User Guide puts together descriptions of user-end parts of the ARC middleware, serving as a user manual and a reference. It also includes basic examples and the ARC client installation instructions.

The User Guide does not provide detailed description of third-party tools, such as the Globus®, giving only minimal necessary information.

The definitive description of the components and services can be found in separate manuals:

User Interface : "The NorduGrid toolkit user interface", user manual [5]

xRSL : "Extended Resource Specification Language" [6]

Grid Manager : "The NorduGrid Grid Manager", description and administrator's manual [3]

Grid Monitor : "The Grid Monitor", usage manual [7]

Detailed installation and configuration instructions are included into the ARC software distribution, available for download at:

`http://www.nordugrid.org`

Any further questions should be addressed to

`nordugrid-support@nordugrid.org`

Chapter 2

Roadmap

On your way to start enjoying the power of Grid computing, you will have to pass the following milestones:

1. **Client**: install the client on your computer, or get access to a computer where the client is installed. See Chapter 3 on client installation.
2. **Certificate**: obtain your Grid passport: a personal certificate. See Chapter 4 on certificates for details.
3. **Access**: become a member of a user group, authorized to use a set of resources (also known as *Virtual Organisation*. See Chapter 5 on Virtual Organisations.
4. **Hands-on**: start a Grid session and try simple tests. See Chapter 6 for guidance.
5. **xRSL**: learn how to describe your task and corresponding requirements in Grid parlance. See Chapter 7 for guidelines.
6. **User interface**: get acquainted with the task and data management on Grid: job submission, status monitoring, cancellation, result retreival etc.. See Chapters 8 and 9 for complete description.
7. **User support**: whenever encountering a Grid problem, write to `nordugrid-support@nordugrid.org`.

Chapter 3

Client Installation

The very first step to get acquainted with ARC-enabled facilities is to download and install the client package. There are several ways to do it, you just have to chose which suits you best. Binary distributions are available for several GNU/Linux flavors, such as RedHat, Mandrake, Debian, SuSE, Fedora, Slackware, etc.. Source distributions are available as well.

The client package may be installed system-wide by a system administrator (Section 3.2), or locally, by a non-privileged user (Section 3.1). A multi-user computer may have several client installations, they do not interfere with each other. A full client takes less than 10 MB of disk space.

You may install a client at any computer you use: e.g., on your office desktop, a notebook, your home computer etc.. You can even load it on a USB memory. One thing to remember when using different clients interchangeably is to keep them in sync, as described in Section 8.2.1.

While client installation procedure itself is trivial, there is a lot of fine tuning that can be done by an advanced user. See Chapter 4 on authorization issues, and Section 3.3 on client configuration.

3.1 Beginner Installation

For a local installation by a user without system administrator privileges, it is advised to download and install a stand-alone distribution of the NorduGrid ARC client. This package is distributed in `.tgz` format and contains all the required third-party components (Globus® components). Scripts provided with the package do all the necessary initial configuration and setup.

3.1.1 Download

Download a pre-compiled stand-alone binary distribution suitable for your operating system from the NorduGrid Downloads area at `http://ftp.nordugrid.org/download` , see Figure 3.1.

3.1.2 Unpack

Put the downloaded package in a directory of your choice and execute

```
tar xvzf nordugrid-arc-standalone-<version>.tgz
```

This will create a new directory `nordugrid-arc-standalone-`<*version*>, and the downloaded `.tgz` package can be safely removed.

3.1.3 Configure

Configure and set up environment variables (most notably, `$NORDUGRID_LOCATION` and `$GLOBUS_LOCATION`) by doing the following:

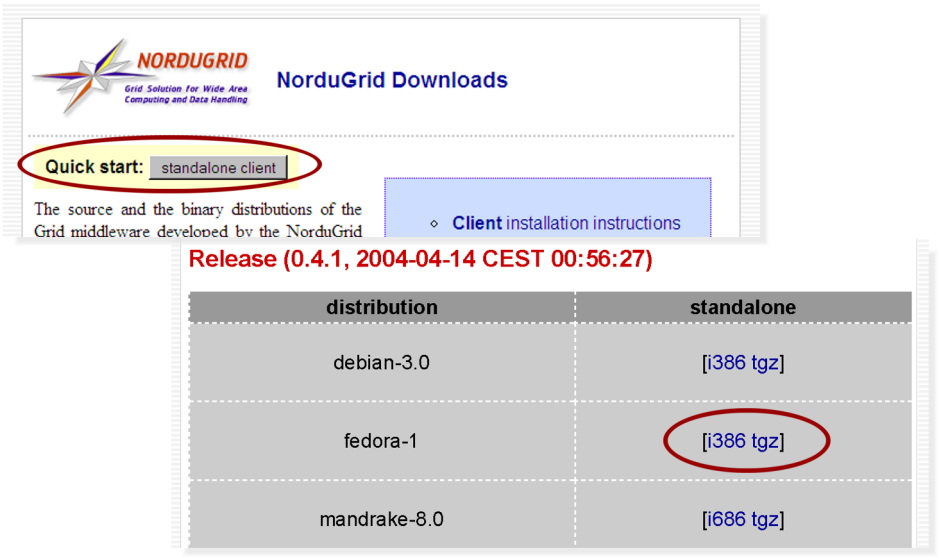

Figure 3.1: Downloading standalone client (example for Fedora Core 1 system).

```
cd nordugrid-arc-standalone-<version>
source setup.sh
```

If you are working in a C shell, use `setup.csh` instead of `setup.sh`. In some environments, you will have to do instead

```
. ./setup.sh
```

in the `nordugrid-arc-standalone-`<*version*> directory. Upon first execution, this will print a lot of informational output. Make sure there are no "error" or "failure" messages.

3.2 System-wide installation

This Section is oriented primarily towards users with system administrator privileges. Local installation by any user is described in Section 3.1.

For a system-wide installation, usage of RPMs is recommended for those Linux distributions which support it. Otherwise, the client can be installed from binary tarballs, or built from the source.

For a fully functioning client, the following packages have to be installed:
- `gpt`
- `globus`
- `nordugrid-arc-client`
- Public keys of your Certificate Authority
- Certificate request configuration files

Installation of the certification-related packages is related to security issues and is discussed in Chapter 4.

3.2.1 Globus Toolkit

The Globus® must be installed at your machine prior to any system-wide Grid installation. You may get it from the Globus project Web site [2], however, it is **strongly** recommended to use the distribution available at the NorduGrid download area, as it contains several critical bug fixes.

Case A. If you already have a Globus® installed on your system, check that the following variables are defined and point to the proper locations:

```
echo $GLOBUS_LOCATION $GPT_LOCATION
```

If the variables are not defined, set them according to your installation. Typically, `GLOBUS_LOCATION` is `/opt/globus` and `GPT_LOCATION` is `/opt/gpt`

Case B. If there is no Globus installation, or you want to profit from the Globus® distribution with the most recent bug fixes and patches, download from `http://ftp.nordugrid.org/download` (see Figure 3.2) and install the necessary packages in the following order:

```
rpm -Uvh gpt-<version>.rpm
export GPT_LOCATION=/opt/gpt (or setenv GPT_LOCATION /opt/gpt for a C-shell)
rpm -Uvh globus-<version>.rpm
export GLOBUS_LOCATION=/opt/globus (or setenv GLOBUS_LOCATION /opt/globus for a C-shell)
```

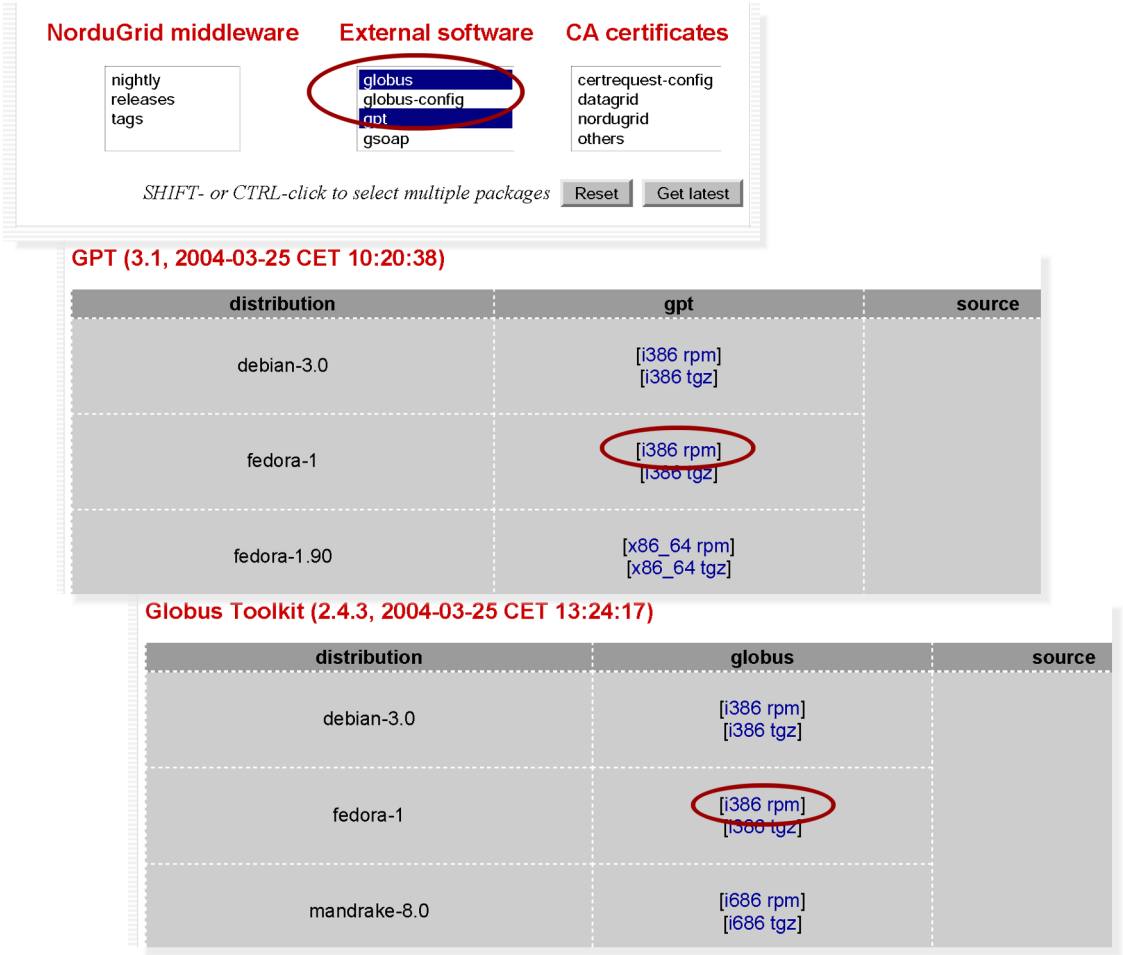

Figure 3.2: Downloading GPT and Globus packages from the NorduGrid Web site. This example shows which RPM packages to download for the Fedora Core 1 Linux distribution.

You may well want to install the Globus® from pre-compiled tarballs provided at the NorduGrid download site. In such a case, take care to execute post-installation procedures:

```
<globus location>/setup/globus-post-install-script <globus location>
```

Here <*globus location*> is the directory to which you unpacked Globus (by default `/opt/globus`).

NB: on some systems and distributions, certain additional external packages need to be installed in order to make Globus Toolkit operational. Some of those are available from the NorduGrid downloads area, "External software" section. See Appendix C for additional information.

3.2.2 Download the client

Download a NorduGrid ARC client package suitable for your operating system and distribution from the NorduGrid Downloads area at `http://ftp.nordugrid.org/download` :

- If you are installing on top of an older Globus version, download a **full source** NorduGrid RPM (`nordugrid-arc-`<*version*>`.src.rpm`) or a tarball (`nordugrid-arc-`<*version*>`.tgz`) (see Figure 3.3).
- If you just have installed Globus as described in Section 3.2.1, download just a binary client RPM `nordugrid-arc-client`<*version*>`.rpm` (see Figure 3.3).

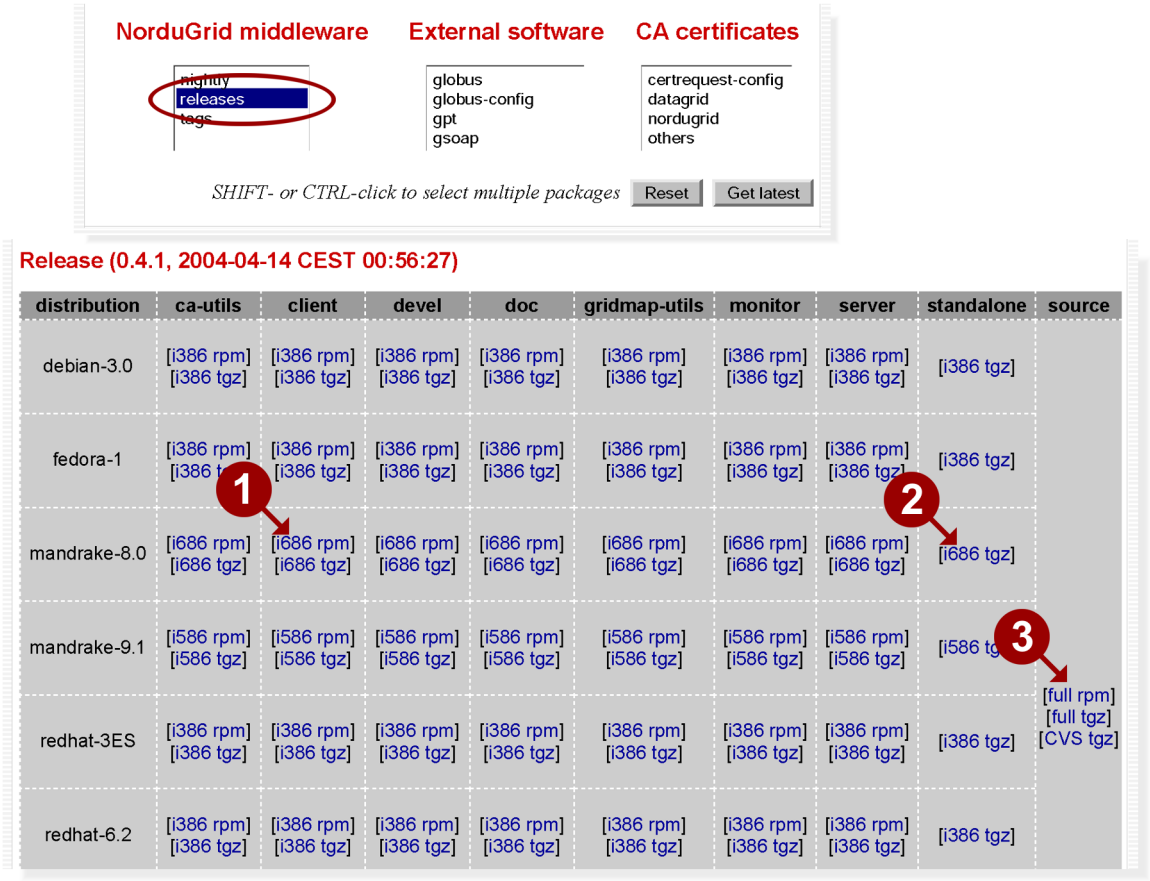

Figure 3.3: Downloading different client packages: (1) binary client RPM, (2) standalone pre-compiled archive, (3) full source RPM. Example shows packages for Mandrake Linux.

3.2.3 Build

If you have installed Globus as described in Section 3.2.1, skip to Section 3.2.4.

If you are installing on top of an older Globus version, you have to rebuild the client from source. Make sure that all the necessary external packages are installed: see Appendix D for details.

When all the external packages are in place, build the NorduGrid ARC from the source RPM:

```
rpmbuild --rebuild nordugrid-arc-<version>.src.rpm
```

This will create several binary RPMs; for the client, you will need only `nordugrid-arc-client`<*version*>`.rpm`.

Alternatively, to build from the tarball, do the following:

```
cat README
tar xvzf nordugrid-arc-<version>.tgz
cd nordugrid-arc-<version>
./configure
make
make install
```

3.2.4 Install the client

Install the client RPM:

```
rpm -Uvh nordugrid-arc-client<version>.rpm
```

Set up the environment variables:

```
source /etc/profile.d/nordugrid.sh
```

Use `nordugrid.csh` for a C shell, or for some other environments one will have to do

```
. /etc/profile.d/nordugrid.sh
```

IMPORTANT! This client will **not** be functional untill you install proper public keys, necessary to establish contact with Grid services. Read Chapter 4 to learn how to do it.

3.3 Client configuration files

3.3.1 client.conf

The default behaviour of ARC client can be configured by specifying alternative values for some parameters in the client configuration file. The file is called `client.conf` and is located in directory `.arc` in user's home area:

```
$HOME/.arc/client.conf
```

If this file is not present or does not contain the relevant configuration information, the global configuration files (if exist) or default values are used instead.

The ARC configuration file consists of several configuration blocks. Each configuration block is identified by a keyword and contains the configuration options for a specific part of the ARC middleware.

The configuration file can be written in either of two different formats: plain text or XML. In the plain text format, client configuration block starts with its identifying keyword inside square brackets: `[client]`. Thereafter follows one or more attribute-value pairs written one on each line in the following format:

```
[client]
attribute1="value1"
attribute2="value2"
# comment line 1
# comment line 2
attribute2="value3"
...
```

In the XML format, the configuration file consists of a single "arc" XML element. Inside this element, client configuration block appears as a separate XML element: `<client>`. This block in turn contains attribute-value elements in the following format:

```
<arc>
 <client>
  <attribute1>value1</attribute1>
  <attribute2>value2</attribute2>
<!-- comment line -->
<!--
commented
block
-->
  <attribute2>value3</attribute2>
  ...
 </client>
</arc>
```

If one has an ARC server installation as well, one can also use the `common` block, which can have the same contents as the `client` one, and is shared with the server.

For multi-valued attributes, several elements or attribute-value pairs have to be specified – one per each value.

The following attributes are allowed:

giis

This attribute is multi-valued.

Specifies which GIISes (site indices) or individual sites to use to discover computing and storage resources. The default is to use the four top level GIISes:

```
ldap://index1.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
ldap://index2.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
ldap://index3.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
ldap://index4.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
```

Examples:

- plain text:

```
giis="ldap://atlasgiis.nbi.dk:2135/O=Grid/Mds-Vo-name=Atlas"
giis="ldap://index1.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid"
giis="ldap://grid.tsl.uu.se:2135/O=Grid/Mds-Vo-name=local"
```

- XML:
```
<giis>ldap://odin.switch.ch:2135/O=Grid/Mds-Vo-name=Switzerland</giis>
<giis>ldap://index1.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid</giis>
<giis>ldap://grid.tsl.uu.se:2135/O=Grid/Mds-Vo-name=local</giis>
```

Note that LDAP URLs containing "Mds-Vo-name=local" refer not to GIISes, but to individual sites. This feature can be used to add "hidden" sites that do not register to any GIIS, or to list explicitly preferred submission sites.

debug

Default debug level to use for the ARC clients. Corresponds to the `-d` command line switch of the clients. Default value is 0, possible range is from -3 to 3.

Examples:

- plain text:
```
debug="2"
```
- XML:
```
<debug>2</debug>
```

timeout

Timeout to use for interaction with LDAP servers, gridftp servers, etc. If a server, during e.g. job submission, does not answer in the specified number of seconds, the connection is timed out and closed. Default value is 20 seconds.

Examples:

- plain text:
```
timeout="20"
```
- XML:
```
<timeout>10</timeout>
```

downloaddir

Default download directory to download files to when retrieving output files from jobs using `ngget`. Default is the current working directory.

Examples:

- plain text:
```
downloaddir="/home/johndoe/arc-downloads"
```
- XML:
```
<downloaddir>/tmp/johndoe</downloaddir>
```

alias

This attribute is multi-valued.

Alias substitutions to perform in connection with the `-c` command line switch of the ARC clients. Alias definitions are recursive. Any alias defined in a block that is read before a given block can be used in alias definitions in that block. An alias defined in a block can also be used in alias definitions later in the same block.

Examples:

- plain text:
```
alias="host1=somehost.nbi.dk"
alias="host2=otherhost.uu.se"
alias="myhosts=host1 host2"
```
- XML:
```
<alias>host1=somehost.nbi.dk</alias>
<alias>host2=otherhost.uu.se</alias>
<alias>myhosts=host1 host2</alias>
```

With the example above, `ngsub -c myhosts` will resolve to `ngsub -c somehost.nbi.dk -c otherhost.uu.se`.

broker

Configures which brokering algorithm to use during job submission. The default one is the `FastestCpus` broker that chooses, among the possible targets, the target with the fastest CPUs. Another possibility is, for example, the `RandomSort` broker that chooses the target randomly among the targets surviving the job description matchmaking.

Examples:

- plain text:
```
broker="RandomSort"
```
- XML:
```
<broker>RandomSort</broker>
```

3.3.2 srms.conf

If any data management commands are used with the Storage Resource Management (SRM) [8] protocol, the file

$HOME/.arc/srms.conf

may be created to store cached information on these services. For more information see the description inside this file.

3.3.3 Deprecated configuration files

In ARC $\leq$ 0.5.48, configuration was done via files `$HOME/.ngrc`, `$HOME/.nggiislist` and `$HOME/.ngalias`.

The main configuration file `$HOME/.ngrc` could contain user's default settings for the debug level, the information system query timeout and the download directory used by `ngget`. A sample file could be the following:

```
# Sample .ngrc file
# Comments starts with #
NGDEBUG=1
NGTIMEOUT=60
NGDOWNLOAD=/tmp
```

If the environment variables NGDEBUG, NGTIMEOUT or NGDOWNLOAD were defined, these took precedence over the values defined in this configuration. Any command line options override the defaults.

The file `$HOME/.nggiislist` was used to keep the list of default GIIS server URLs, one line per GIIS (see `giis` attribute description above).

The file `$HOME/.ngalias` was used to keep the list of site aliases, one line per alias (see `alias` attribute description above).

Chapter 4

Grid Certificates

This Chapter is not an easy reading, but the presented knowledge is essential for newcomers unfamiliar with Grid. Impatient readers can go to Section 4.1 for quick instructions; however, it is strongly recommended to eventualy read the entire Chapter.

In a Grid environment such as the NorduGrid/ARC, users normally don't have local password-protected accounts on computing resources they intend to use. You should hold instead an **electronic certificate**, which ensures unique authentication.

This Chapter gives some basic information on:

a) what is a **Certificate Authority** and how to affiliate yourself with an appropriate one
b) how to request **user certificates**
c) how to use a certificate
d) why do you have to trust different Certificate Authorities and how to obtain their credentials

Possession of a certificate, however, does not automatically authorize you to use any Grid resource.

Access control for the computing resources is a matter of local policy: site administrators retain full control of choosing which Grid user is allowed to use their resources. Typically, such local authorization process is done by mapping an accepted set of Grid users onto a set of local user accounts. See Chapter 5 on how to get access to the Grid resources and other related information.

4.1 Quick start with certificates

This section is primarily intended for the Nordic countries (Denmark, Finland, Iceland, Norway and Sweden) and Estonia residents (current members of the NorduGrid collaboration). Users from other places should modify instructions accordingly or read further for detailed information.

The amount of setup you have to do regarding certificates depends on the way you installed the client.

4.1.1 Certificates in local standalone client

Luckily, the standalone client (Section 3.1) comes with all the necessary pre-requisits. Continue to Section 4.1.3 to learn how to obtain a personal certificate.

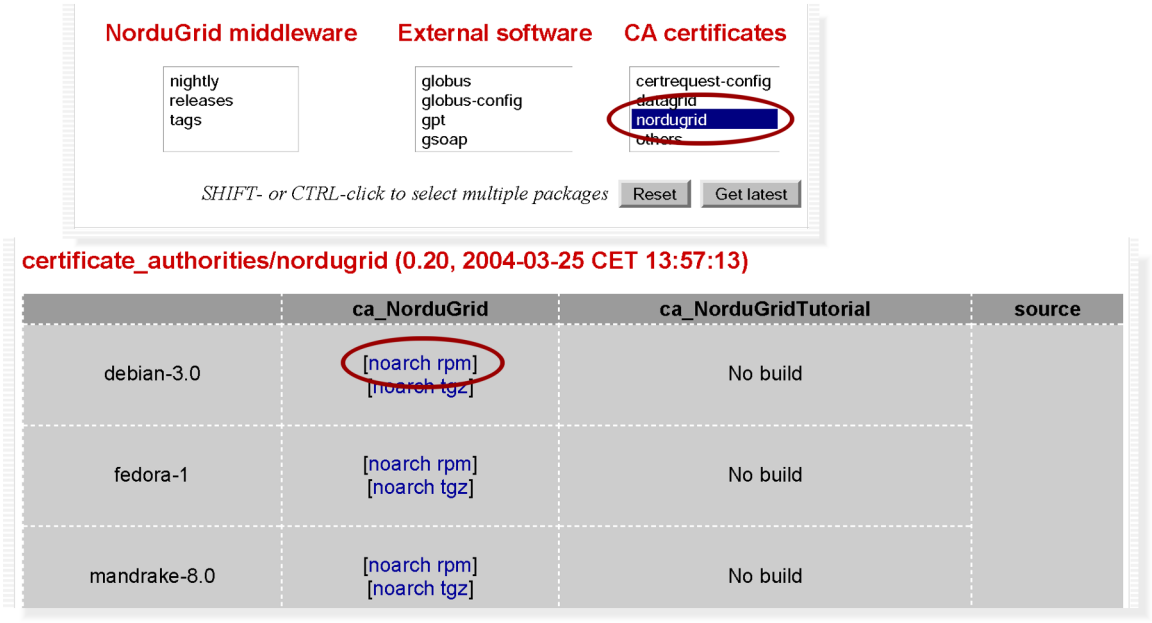

Figure 4.1: Downloading NorduGrid Certificate Authority public keys. Example shows packages for Debian Linux.

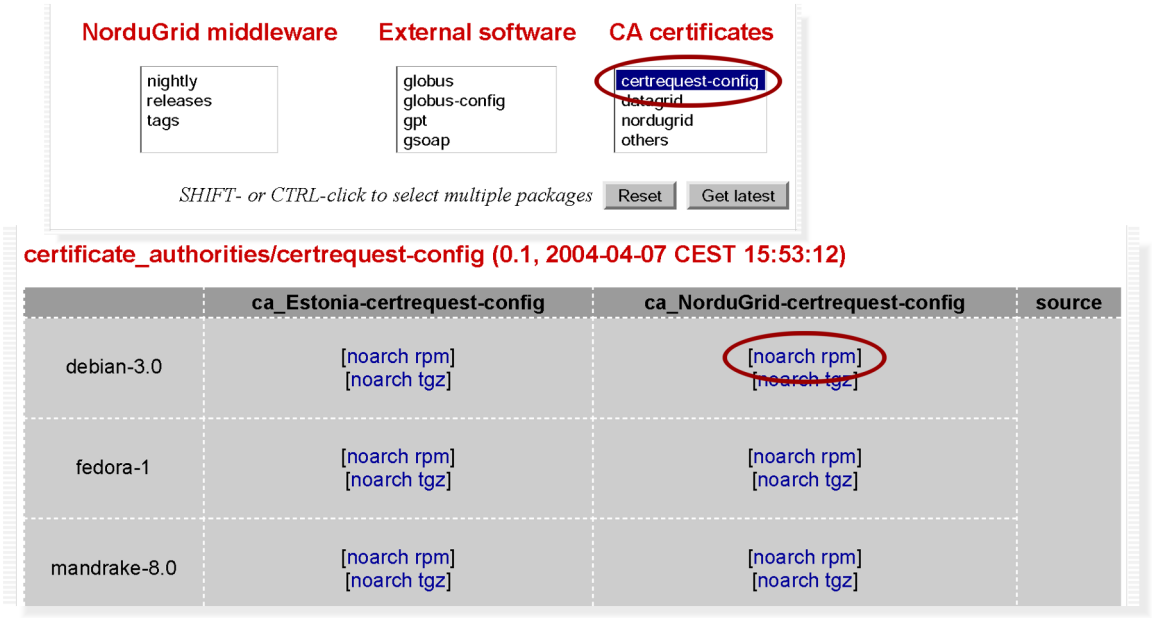

Figure 4.2: Downloading NorduGrid Certificate Authority certificate request configuration package. Example shows packages for Debian Linux.

4.1.2 Certificates for system-wide client installation

System-wide client installation from RPMs or tarballs, as described in Section 3.2, does **not** include files (public keys and configuration templates) necessary for certificate request and usage operations. Therefore, you have to obtain them from the relevant Authority (see Section 4.2 for information on different Certificate Authorities). Examples in this Section refer to the NorduGrid one*.

> You will have to download and install two packages:
> `ca_NorduGrid`
> `ca_NorduGrid-certrequest-config`

The packages are available as RPM or tarball files from the NorduGrid download area at `http://ftp.nordugrid.org/download` , in the "CA certificates" section. You will need `nordugrid` certificates and `certrequest-config` package, see Figures 4.1 and 4.2 for illustration.

Install the downloaded packages; e.g., for RPMs:

```
rpm -Uvh ca_NorduGrid*
```

You have now all the necessary requisits to request a NorduGrid certificate and to use it for your authentication. Continue to Section 4.1.3 to learn how to request a personal certificate.

4.1.3 Obtain a personal certificate

To obtain a personal certificate, do:

```
grid-cert-request -int -ca
```

Follow **carefuly** the instructions:

1. Select the appropriate Authority from the list: NorduGrid (hash 1f0e8352) or Estonia (hash 566bf40f)
2. Enter the password (called "passphrase") and **memorize** it
3. If asked, enter the two-letter country code (`EE` for Estonia) or hit "Return"
4. Enter `Grid` for Level 0 Organization Name or hit "Return"
5. If asked, enter `NorduGrid` for Level 1 Organization Name or hit "Return"
6. Enter your domain: typically, the string which follows `@` in your official e-mail address, e.g. `cd.uit.fi`
7. Enter your name **using strict Latin alphabet characters**, e.g. `Goran Mueller`
8. Enter your official e-mail address

This procedure will create a new directory called `.globus` in your home directory, and generate several files there. If you already have this directory, make sure it is empty or rename it **before** issuing `grid-cert-request`, e.g., to `.globus.old`.

Send the certificate request to the corresponding Certificate Authority, NorduGrid CA if you are a Nordic country resident:

*Same instructions are valid for the Estonian certificates, replace simply "NorduGrid" with "Estonia" in all the strings

```
cat ~/.globus/usercert_request.pem | mail ca@nbi.dk
```

or the Estonian CA, if you live in Estonia:

```
cat ~/.globus/usercert_request.pem | mail ca@grid.eenet.ee
```

Now you only have to wait: within 2 working days you will receive an e-mail containing your public certificate, signed by the proper authority. Find the part of the message that looks like:

```
-----BEGIN CERTIFICATE-----
MIIC6zCCAlSgAwIBAgICAPAwDQYJKoZIhvcNAQEEBQAwTzENMAsGA1UEChMER3Jp
ZDESMBAGA1UEChMJTm9yZHVHcmlkMSowKAYDVQQDEyFOb3JkdUdyaWQgQ2VydGlm
aWNhdGlvbiBBdXRob3JpdHkwHhcNMDMwNTIxMTkzOTA3WhcNMDQwNTIwMTkzOTA3
WjBQMQ0wCwYDVQQKEwRHcmlkMRIwEAYDVQQKEwlOb3JkdUdyaWQxEjAQBgNVBAsT
MFIxDTALBgNVBAoTBEdyaWQxEjAQBgNVBAoTCU5vcmR1R3JpZDEUMBIGA1UECxML
cXVhcmsubHUuc2UxFzAVBgNVBAMTDk94YW5hIFNtaXJub3ZhMIGfMA0GCSqGSIb3
DQEBAQUAA4GNADCBiQKBgQC5qAKyKGvO/6G7VC2CnICXWAwvKFX961AcyKeT5zxg
lDXUVxlhrLnHGKh9tlUZxG0NCT1UTQ+W4FVo5EPSpdsHI1MOyoVUF+HNIgII/TNj
AgMBAAGjgdQwgdEwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBH
ZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFLmvMBnYA8BjeeOmtDzmE84y
2fCiMHcGA1UdIwRwMG6AFBgFwPwL0bc69GWSCftZoV/HiMTwoVOkUTBPMQ0wCwYD
VQQKEwRHcmlkMRIwEAYDVQQKEwlOb3JkdUdyaWQxKjAoBgNVBAMTIU5vcmR1R3Jp
ZCBDZXJ0aWZpY2F0aW9uIEF1dGhvcml0eYIBADANBgkqhkiG9w0BAQQFAAOBgQC1
VPU8sBArpPAoZKgWOmM7C7EZBQ2xKMAU+uJqWcAHPqGEReGbghCBIwIsQP3onT2Y
SqFWQSHM5gXBZ6sWGw+YrAiGfURgsSsYZwSa9Bs6mJcANxNU8HioyY1ngJwVJMDr
lxcO0tWbHZpekWMe9QG8gn/OUKdXz1TKUS9Cx+mBZw==
-----END CERTIFICATE-----
```

Save such text in

```
~/.globus/usercert.pem
```

Check the information stored in the certificate:

```
grid-cert-info
```

Now you can use this certificate as your electronic passport on the Grid and elsewhere. Read Chapter 5 and apply for access to Grid resources, and then go to Chapters 7 and 8 to learn how to work on Grid.

4.2 Grid Authentication And Certificate Authorities

Instructions given in Section 4.1 refer to the NorduGrid Certificate Authority as the body that is appointed to issue Grid certificates in Nordic countries. This section describes basics of Grid security and explains what are Certificate Authorities, what are their functions, and what you should do in a generic case in order to obtain proper certificates.

4.2.1 Grid Login, Proxies

One of the main purposes of the Grid is to provide a transparent access to computing and storage resources. For a user, it means that a single login procedure must give access to all the available Grid resources.

Technically, this is implemented via a system of proxies. A user issues a temporary public proxy, which contains encrypted information about the issuer, and allows Grid services to act on her behalf while the proxy is valid. Thus, when a user wants to contact a resource, her client upon request presents the proxy to the remote server, which saves user from typing in login name and password.

Any Grid operation requires a valid proxy issued by the user. Validity of proxies is time-limited; typically, they expire in 12 hours, although users can issue proxies of any lifetime.

To make such handshake between the user's client and a Grid service happening, the following conditions must be satisfied:

- the remote server must be able to check the proxy authenticity;
- the user (her client) in turn must be able to check the remote server authenticity through its credentials.

The Grid utilizes public key, or asymmetric cryptography, for authentification of users, resources and services [9]. According to the basics of the public key cryptography, each actor on the Grid (a user or a service) must have a key pair: a **public** and a **private** key. The public key is, quite naturally, made public while the private key must be kept secret.

AVOID STORING YOUR PRIVATE KEY ON A PUBLIC FACILITY, SUCH AS A SHARED ACCESS COMPUTER, KEEP IT ON YOUR PRIVATE COMPUTER INSTEAD. COMPROMISED KEYS ARE REVOKED BY THE AUTHORITY.

Encryption and authorization is performed using the public key, while decryption and digital signature is performed with the private key.

Since Grid services act on behalf of users, such services and all the resources which provide them must have a key pair as well.

4.2.2 Certificate Authorities

Keys are generated by users or service administrators with the help of standard tools (see, e.g, Section 4.2.7). In order to become accepted by other parties, public keys must be *signed* by trusted authorities of the Grid, called **Certificate Authorities** (CA).

It is important to note that generating a key pair does not automatically provide you access to the Grid resources. A CA needs to sign your key pair, thus confirming your identity. This signing procedure of the CA is often referred to as *"issuing a certificate"*.

Issuing a certificate is similar to issuing a national passport to a person, and typically, there is one CA per country. There are few exceptions though: for example, NorduGrid CA serves all the Nordic countries, while in bigger countries like USA, France or Germany, there are ceveral CAs. For a non-authoritative list of CA's, refer to Appendix E.

The role of CAs is not only to confirm users identities and certify resources, but also to keep track of abuse of the issued certificates. CA's regularly publish Certificate Revocation Lists (CRL), with which all the Grid services comply. If your private key happened to be in possession of another person, or your identity was used for illegal exploits, your certificate is revoked and you will be unable to work on Grid. The same is true for resource certificates: if a computer is used for repetitive offences, its certificate will be revoked.

4.2.3 Globus Grid certificates

In the Globus framework, the key pair of public-key cryptography consists of the key file (default name `userkey.pem`) and the public certificate file (default name `usercert.pem`). This means every Grid user must have these two files every time she has to communicate with Grid resources.

The `userkey.pem` file (or `resourcekey.pem` when a Grid resource is concerned) contains the private key encrypted with a password (called *"pass phrase"* by Globus).

> The *"pass phrase"* is a password to protect a certificate. Any certificate, be it a user, a host, or a service one, must be protected by a separate password. **DO NOT** confuse it with a user or a system administrator log-in password!

The certificate file `usercert.pem` contains your public key together with additional important information such as the *Subject Name* (SN) of the certificate holder, the name of the signing CA, and a digital signature of this CA. The important role of the CA is to establish a trustful connection between the identity of the user and the public key in the certificate file. The digital signature of the CA in the user's certificate file officially declares that the public key in the file indeed belongs to the specific user (Subject Name). This is similar to the police department stamp in your passport: it confirms that the document indeed contains correct information.

The certificate files are encoded with the x.509 [10] format and need special tools to get decrypted. See Section 4.2.7 for some examples.

4.2.4 Associating yourself with a proper CA

To generate a certificate request, you must know to which CA this request will be submitted. The process is similar to a passport application: you must chose to which country (authority) to apply, and that authority will decide whether to grant your request or not.

Many countries have a CA, or even several – depending on the area of activities these authorities cover. Some countries have one CA for all – like the Nordic countries. Yet most countries have no CA, and citizens of such have to establish a CA and make sure it complies with intrenational rules and is consequently accepted by others. Meanwhile, the French Committee for Scientific Research (CNRS) issues certificates for most academic and edcational users, but this is a provisional arrangement. The list of some known CAs can be found in Appendix E. Unfortunately, there is no commonly accepted world-wide body that keeps track of and validates various CAs.

In case you are a Nordic country resident and have no certificate, you should request it from the NorduGrid Certificate Authority. To do this, you should download at `http://ftp.nordugrid.org/download` (section "CA certificates", see Figure 4.2) two packages:

1. Public keys of the NorduGrid Certificate Authority
2. The corresponding `certrequest-config` configuration package

The standalone ARC client installation keeps keys and configuration files in the directory `$NORDUGRID_LOCATION/share/certificates` , which by default contains all the files necessary for the Nordic users.

If you use a centralized ARC installation, install the RPM packages using

```
rpm -Uvh ca_NorduGrid-*
```

or use the provided tarballs. The following files will be installed in `/etc/grid-security/certificates`:

```
1f0e8352.0
1f0e8352.crl_url
1f0e8352.signing_policy
globus-host-ssl.conf.1f0e8352
globus-user-ssl.conf.1f0e8352
grid-security.conf.1f0e8352
```

You may well prefer installing the files in any location other than `/etc/grid-security/certificates`, and point the `$X509_CERT_DIR` variable to such alternative location.

If you reside in any other country, please contact your local CA or the CNRS one (see Appendix E) for public keys and configuration files. They will have the names similar to those shown in the example above, except for the alphanumeric 8-characters hash. Install them either in `/etc/grid-security/certificates`, or in any other location pointed by the `$X509_CERT_DIR` variable.

If you are using the standalone client, the files must be installed in `$NORDUGRID_LOCATION/share/certificates` or in any other location pointed by the `$X509_CERT_DIR` variable.

You can also install all the necessary keys and configuration files in your private `$HOME/.globus/certificates/` directory; in this case however you will not be able to use those einstalled centrally.

You may have as many keys and configuration files installed as you wish. Moreover, you **must** have the public keys of all the CAs that certify the Grid resources you will use, and of all the CAs that certify potential users of your own resource. Section 4.2.5 has some details on such "Friendly CAs".

Please remember that installing a CA key implies that you accept their rules and policies, and trust their judgement of users, – in other words, you establish a trust relationship with the authority.

4.2.5 Friendly CAs

Depending on what is your Grid certificate issuing authority and which Grid resources you intend to use, you will have to install public certificates and other attributes of the necessary authorities. Table 4.1 presents a short check-list, summarizing which certificates you would need.

Your certificate	*Will use NorduGrid resources*	*Will use other Grid resources*
NorduGrid certificate	NorduGrid CA certificate and signing policy	Certificates, signing policies and other attributes (if any) of all the CAs that certified the resources
Other Grid certificate	Your issuing CA certificate and signing policy file (if any) and the NorduGrid CA certificate and signing policy	
No Grid certificate	NorduGrid CA certificate and signing policy, and certificate request configuration files	

Table 4.1: Check-list of needed certificates

The recommended procedure to obtain other CA certificates and attributes is to fetch them directly from the CAs in question. They are typically available from their respective Web sites (for a non-authoritative list, see Appendix E) .

If for some reason it is impossible to fetch your CA attributes (e.g., Web site unavailable, holiday season *etc.*), you may download the necessary files from the NorduGrid downloads area at `http://ftp.nordugrid.org/download` . The attributes are platform- and system-independent. Install the CA attributes either from RPMs:

```
rpm -Uvh <your CA name>*
```

or from the provided tarballs.

IMPORTANT! The rule of thumb when installing the NorduGrid ARC middleware is **a**lways to check the contents of the `/etc/grid-security/certificates` and/or `$X509_CERT_DIR` folder and to make sure that it does contain certificates and signing policy files of your certificate issuer CA and the friendly CAs. A typical listing of the folder should look as follows:

```
> ls -1 $X509_CERT_DIR
1f0e8352.0
1f0e8352.signing_policy
bc870044.0
```

```
bc870044.signing_policy
d64ccb53.0
```

Here 8-symbol file names with extension ".0" are the certificates containing public keys of different CAs.

Different users may trust different CAs, not necessarily those centrally accepted. In such a case, they should collect the keys of CAs they trust either in the `$HOME/.globus/certificates/` directory, or in any other place they define as `$X509_CERT_DIR`. In such a case, Grid utilities will look for the keys **only** in the user-specified directory.

If you have a `$HOME/.globus/certificates/` directory, make sure it contains your own CA key, as well as other friendly CA keys. If your central system installation is missing a key, you always can install it in your `$HOME/.globus/certificates/` directory, but then you **must copy all the other needed keys** to this directory.

In general, the certificates and configuration files can be located in any of the following directories:

```
/etc/grid-security/certificates
$NORDUGRID_LOCATION/share/certificates
$X509_CERT_DIR
$HOME/.globus/certificates
```

The latter has the precedence, as each user is free to chose whom to trust.

4.2.6 Certificate Request

In order to obtain a valid "passport" on the Grid, you need to create a key pair and submit your public key to your local CA to get an approval. This process is called a *"certificate request"*. The CA will follow its certificate policy and upon successful evaluation of your request your public key will be signed and sent back to you. This may involve communication between the CA and your employer, or any other adequate procedure.

As it was mentioned before, all the resources (i.e. gatekeepers, storage facilities or other services) require a CA-signed key pair as well, in order to be able to operate on the Grid. In this case, resource owners create host key pairs and send them to be vaidated by the local CA. This is similar to registering a car and obtaining a license plate.

Please always remember that a Grid passport consists of two files, the **private key** file and the **public certificate** file. You need to have both of them, the certificate file (`usercert.pem`) alone is not enough for the Grid. If you loose one of your key files, you will have to generate a new CA-signed key pair.

To request a personal or a host certificate, you must decide to which authority you want to submit such a request. Appendix E lists some known authorities and their contact addresses. Each CA has its own procedure, which you have to follow. Typically, you must request a certificate from the CA of your country of residence. If your country has no CA, you may ask the CNRS "catch-all" CA to provide you with one.

If you are not a Nordic country resident, you may well try (upon consulting the NorduGrid personnel at `nordugrid-support@nordugrid.org`) to use the procedures described here to request a certificate from another Certificate Authority. However, most of them have own established procedures (see Appendix E for a reference), and you are **strongly** advised to follow them.

Before requesting the certificate, **please make sure** that your CA public keys and configuration files are installed as described in Section 4.2.4.

The certificate request should be prepared by doing the following:

```
grid-cert-request -int -ca
```

From the presented list, select the appropriate CA (for example, NorduGrid, if you are a Nordic country resident), and then enter the requested information at each interactive prompt. Typically, you should use the suggested default values, and your proper data: domain, name, e-mail etc. Upon request, type and confirm your certificate protection password.

> NOTE: the certificate protection password is not related to any other password, and must be different from, e.g., your system login password. **Memorize well** the password, as it can not be reset! Although you can change the password any time (see Section 4.2.7), if you forget the password, you will have to request a new certificate.

Follow the printed suggestions strictly, e.g., for the Nordic certificate, do the following:

```
cat $HOME/.globus/usercert_request.pem | mail ca@nbi.dk
```

Alternatively, simply send the file `$HOME/.globus/usercert_request.pem` to the contact e-mail address of your CA (`ca@nbi.dk` in case of a Nordic certificate).

For detailed step-by-step instructions, check Section 4.1.3.

The certificate request procedure creates the new directory `.globus` in your `$HOME` area, and places there three files[†]:

`usercert_request.pem`	the official certificate request, to be mailed to your CA
`userkey.pem`	your **p**rivate key
`usercert.pem`	your public certificate placeholder (initiated with zero size)

The `userkey.pem` holds your private key encoded with your pass phrase (you are prompted to supply this pass phrase during the key pair generation). This file **must only be readable by its owner**, that is, you. The `usercert_request.pem` file contains your unsigned public key together with your Subject Name and the name of your CA. The `grid-cert-request` creates also an empty `usercert.pem` file, just as a placeholder, which later must be overwritten with your CA-signed certificate that you will receive by e-mail.

When and if the CA fulfils your request, they will send you the actual public certificate, which you must store as `$HOME/.globus/usercert.pem`, overwriting the placeholder.

You can use the `openssl` [11] cryptography toolkit and the Globus provided commands to create, check and convert between different formats, and to manipulate your certificate files (actually, the Globus commands are just a friendly interface to the `openssl` toolkit). For further information, please read the `man` pages for `openssl`, `verify` and `x509` commands, or use the Globus commands with the `-help` option. Section 4.2.7 describes some most often used commands.

4.2.7 Working with certificates: examples

Certificate request: the following command creates a key pair for a user or a gatekeeper and prepares a formal certificate request

```
grid-cert-request -int
```

[†]Keys and certificates, as well as their locations, can have different names. In the instructions above, the default naming scheme was used. If you would like to use different names, feel free to contact `nordugrid-support@nordugrid.org` for further directions.

Change pass phrase of the certificate: use this command to change the pass phrase (password) of the private key file `userkey.pem`:

```
grid-change-pass-phrase -file userkey.pem
```

> Please note that once you forget the pass phrase, the certificate has to be requested anew, as there is no way to re-generate a password.

Inspect a certificate: to print all the information from the public certificate file `usercert.pem`, do the following

```
grid-cert-info -file usercert.pem -all
```

In particular, the line starting `Subject:` contains your certificate Subject Name (SN), that has to be communicated to a VO manager or a site administrator in order to get access to the Grid (see Section 5).

Duplicate a certificate with a new pass phrase: using this tool, your private key is encoded with a new pass phrase and stored in the `new_userkey.pem` file (first it asks for your old pass phrase, then twice for the new).

```
openssl rsa -in userkey.pem -des3 -out new_userkey.pem
```

Verify the certificate: to verify the `usercert.pem` certificate using the public key of the issuing CA, which is supposed to be in located in the specified `CApath`, do this:

```
openssl verify -CApath /etc/grid-security/certificates/ usercert.pem
```

Dump contents of the certificate: with this command you can display the contents of the `usercert.pem` certificate

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

Convert your Grid certificate into a Web one: you can use your Grid certificate to to authenticate yourself on the Web via Internet browsers. Before importing your certificate into a browser, you must convert it from the original `pem` format to `pkcs12` one:

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

Here `cert.p12` can be any name you like (preferably with the `.p12` extension). This file can be imported into any modern browser and serve as a Web cerificate. Use your Grid passphrase to manage it.

Chapter 5

Getting Access to Grid Resources: Virtual Organisations

NB: possession of a Grid certificate, even signed by a legal CA, does not automatically allow you to use any Grid resources!

As it was mentioned above, in order to be able to do such things as copy files over the Grid, or submit Grid jobs, your identity should be mapped to a local account on each resource – be it a computing cluster or a storage facility. As long as this mapping is done, you are *authorized* to use a resource as any local user.

5.1 NorduGrid Virtual Organisation

To facilitate and automate such mapping for the Nordic Grid users, NorduGrid has set up a collective authorization method, the NorduGrid Virtual Organization (VO),, following practices of other Grid testbeds. This VO maintains a list of people which are authorized to use the Nordic Grid resources. The VO tools provide an automatic method for the sites to easily maintain the *"VO member"↔"local Unix user"* mappings. These tools periodically query the VO user database and automatically generate the local *grid-mapfiles* following the local policy formulated in the VO-tools configuration file. The automatic mapping does not violate the site autonomy, because the site administrators retain a full control over their systems thanks to the possibility of denying access to "unwished" Grid users in the NorduGrid VO-tools configuration file.

The database of the VO is maintained by the *VO managers*. Their responsibility is to add, delete or modify user entries.

The NorduGrid VO is limited to the users affiliated with the Nordic organisations. To become a member, you must send a request containing your certificate Subject Name (see Section 4.2.7) to `nordugrid-support@nordugrid.org`. If you are not a Nordic countries resident, you may be added to the *Guests* VO.

The NorduGrid VO supports the creation of groups. A group is a subset of the a VO and is maintained by an appointed group manager. The group manager selects members of the group out of the full VO database. With the existence of the user groups, the site administrators can implement group-based mappings, such that all the members of a certain group are mapped to the same local Unix user, in addition to the default user-based mappings.

Technically, the VO database is stored in an LDAP [12] database. For this purpose, a GSI [13] enabled OpenLDAP server is used, providing an entry and attribute level access control, based on the Grid certificates. The database managers, being authenticated and authorized through their certificates, make use of the OpenLDAP command line tools in order to add, delete or modify entries in the VO database. The Nordic Grid sites periodically (4 times a day) run the `nordugridmap` utility in order to query the VO LDAP

server and automatically create/update local user mappings according to a site policy (as defined in a `nordugridmap.conf` configuration file).

5.2 Other Virtual Organisations

Except of the core Nordic VO, NorduGrid maintains a *Guests* VO, similar in organisation and implementation to the NorduGrid one. Every user willing to try NorduGrid/ARC facilities can be added to the Guests VO upon contacting `nordugrid-support@nordugrid.org`.

Many other VOs are operating on the Grid, typically consisting of people involved in the same research project. Each such VO has a manager who maintains the database and negotiates resources available for the members. It is recommended for every user to join a VO in order to get access to the Grid.

In case you do not belong to or do not qualify for any existing VO, you always can establish your own VO and be your own manager. You will have to contact the Grid resource owners one by one and negotiate resource allocation with them. Feel free to contact `nordugrid-support@nordugrid.org` for initial instructions or possible assistance.

Chapter 6

Grid Session

Before starting your first Grid session, check whether the following have been accomplished:

1. You have the ARC client installed (see Chapter 3)
2. You have a valid Grid certificate (see Section 4)
3. You are authorised at at least one Grid site (see Section 5)

6.1 Logging Into The Grid

Access to the Grid resources in the Globus world is made via a so-called *proxy* – a kind of a temporary token, which you pass to the Grid services, allowing them to act on your behalf.

Initialization of the proxy is done via the standard Globus command:

```
grid-proxy-init
```

This command searches for your public certificate and private key in the default directory (`$HOME/.globus`), and upon a pass phrase confirmation, creates a file containing your Grid proxy in the `/tmp` directory of your computer. The proxy file name typically starts with `"x509up_u"`, which is followed by your UNIX/Linux UID.

Your Globus proxy is public, hence anybody having access to it can impersonate you. To minimize this risk, proxies have limited lifetime. By default, a proxy is valid for 24 hours. This, however, means that if your Grid task will last longer than the proxy validity, it will not be able to finish correctly because of the proxy expiration. Either for this reason, or, on the contrary, if you'd like to have your proxy short-lived, it is possible to create a proxy with a user-defined lifetime:

```
grid-proxy-init -valid 27:45
```

In this example, a proxy will live 27 hours and 45 minutes.

The `grid-proxy-init` command has several options: e.g., you can select a location of the newly generated proxy, or specify another location for your Grid certificate and/or key. For detailed information on these options, use `grid-proxy-init -help`.

If you are using different computers to log in to the Grid, it is **strongly** advised **NOT TO COPY** your private key across the systems. Instead, use a dedicated machine to generate a proxy, and then copy the proxy file to another computer, preferably into your `$HOME` directory, and describe the new proxy location in the `X509_USER_PROXY` environment variable:

```
scp /tmp/x509up_u230 another.machine.org:myproxy
ssh another.machine.org
export X509_USER_PROXY=$HOME/myproxy
```

6.1.1 Proxy Handling Tips

> To avoid troubles, please always try to make sure that your proxy has a validity period sufficient for your task to come to completion. However, it is not advised to generate proxies with lifetimes of several days.

6.2 First Grid test

ARC comes with a test suite that allows users to quickly test their installation and most of the Grid functionality. The very first test can be performed by issuing

```
ngtest 1
```

This will produce some informative output, and upon a successfull test job submission will print out the job handle, or *job ID*, which may look like:

```
gsiftp://site.it.uni.org:2812/jobs/10308913211503407485
```

This will mean that your test job is sent for execution to the site `site.it.uni.org` and the output is available via this `gsiftp://` URL. `gsiftp` is the Grid analogue of FTP, and uses your proxy instead of login/password authentication.

To see what is the status of the job, do

```
ngstat gsiftp://site.it.uni.org:2812/jobs/10308913211503407485
```

A typical output will look like:

```
Job gsiftp://site.it.uni.org:2812/jobs/10308913211503407485
Jobname: ngtest-job-1
Status: ACCEPTED
```

You can check what the job sent to the standard output by doing

```
ngcat gsiftp://site.it.uni.org:2812/jobs/10308913211503407485
```

If `ngcat` displays the words `hello,grid`, the first test has completed successfully! You can now clean up by deleting all the job traces:

You can check what the job sent to the standard output by doing

```
ngclean gsiftp://site.it.uni.org:2812/jobs/10308913211503407485
```

For more test options, check

```
ngtest --list
```

All the job manipulation commands (`ngstat`, `ngcat`, `ngclean` etc) can with a bulk of jobs, with a specific cluster, and otherwise have many other options. Check Section 8 for the complete description.

To learn how to describe and submit your own job, proceed to Section 7.

6.3 Logging Out

Logging out is performed by destruction of the proxy. You can either physically erase the corresponding file, or issue

```
grid-proxy-destroy
```

Chapter 7

Describing Grid Tasks

To describe a task to be submitted to the NorduGrid/ARC resources, a special scripting language is used. It is strongly based on the *Resource Specification Language (RSL)*, developed by the Globus project [14]. RSL allows job options and definitions to be parsed to resource management systems. The ARC architecture requires certain extensions to the RSL. This concerns not only introduction of new attributes, but also differentiation between two levels of the job options specifications:

User-side RSL, i.e., the set of attributes specified by a user in a job-specific file. This file is interpreted by the *User Interface (UI)* [5], and after the necessary modifications is passed to the *Grid Manager (GM)* [3]

GM-side RSL, i.e., the set of attributes pre-processed by the UI, and ready to be interpreted by the GM

As a user, you have to know only the user-side part, and utilize it to describe your Grid tasks.

In what follows, the description of the NorduGrid-extended RSL, further denoted as **xRSL**, is given, using the following notations:

`<xxxx>`	parameter to be substituted with a corresponding string or a number
`[xxxx]`	optional parameter
`xxx|yyy|zzz`	list of possible values of a parameter
`-"-`	"same as above"

7.1 Task Description Language: xRSL

For a complete description of Globus RSL, see reference [14]. xRSL uses the same syntax conventions, but changes the meaning and interpretation of some attributes.

A Grid task is described by the mean of xRSL attributes, which can be either passed via a command-line, or, more conveniently, be collected in a so-called xRSL-file (suggested extension *.xrsl*). Such a file contains a plain list of attribute assignment strings (*relations*) and boolean operands "&" (for AND) and "|" (for OR). Attribute names are case-insensitive.

An attribute-value pair is a key element of task description. It consists of a **relation** that assigns one or more values to an attribute, and is enclosed in round brackets:

```
(attribute="value")
```

An attribute can have several values (blank-separated list):

```
arguments="abc" "123" "dir/file"
```

or even correspond to pairs of values:

```
inputFiles=("thescript" "http://my.site.org/taskX/script1.sh")
           ("my.idx" $HOME/index/the_index.bin)
           ("thescript" ""))
```

In the examples above, some strings are quoted, while some are not. When explicitly quoted, a string may contain any character. However, most literals used in job description (e.g. attribute names themselves, file names etc) don't have to be quoted, if they don't contain **special characters** and blank spaces.

The special characters are:

```
+   &   |   (   )   =   <   >   !   "   '   ^   #   $
```

To quote a string containing special characters, you can use either single or double quotes. If your string, however, contains both such characters, you can define any character as an own delimiter, by preceding it with the "carat" (^) character: `jobName=^*My "good" job^*` makes use of a carat-escaped asterisk as a delimiter.

Typically, an xRSL job description starts with an ampersand ("&") , to indicate implicit **conjunction** (boolean `AND` style) of all the attribute relations, enclosed in parentheses:

```
&(attribute1=value1)(attribute2="value 2")...
```

Whenever a **disjunct**-request (boolean `OR` style) of two or more attributes is needed, the following construction can be used:

```
(|(attribute="value1")(attribute="value2")...)
```

In expressions, the following operands are allowed:

```
=   !=   >   <   >=   <=
```

Commented lines should start with "(*" and be closed with "*)":

```
(*attribute="value1"*)
```

Comments can not be nested.

Multiple job descriptions in one file are realized via a standard Globus RSL multi-request operand "+", which should precede the multiple job descriptions:

```
+(&(...))(&(...))(&(...))
```

The xRSL attributes can be written in a single string, or split in lines arbitrary; blank spaces between `(attribute="value")` pairs and inside relations are ignored.

7.1.1 URLs

File locations in ARC can be specified both as local file names, and as Internet standard *Uniform Resource Locators (URL)*. There are also some additional URL *options* that can be used.

The following transfer protocols and metadata servers are supported:

`ftp`	ordinary *File Transfer Protocol (FTP)*
`gsiftp`	GridFTP, the Globus® -enhanced FTP protocol with security, encryption, etc. developed by The Globus Alliance [2]
`http`	ordinary *Hyper-Text Transfer Protocol (HTTP)* with PUT and GET methods using multiple streams
`https`	HTTP with SSL v3
`httpg`	HTTP with Globus® GSI
`ldap`	ordinary *Lightweight Data Access Protocol (LDAP)* [12]
`srm`	Storage Resource Manager (SRM) service [8]
`se`	ARC Smart Storage Element service [15]
`lfc`	LFC catalog and indexing service of EGEE gLite [16]
`rc`	Globus® *Replica Catalog (RC)* [17]
`rls`	Globus® *Replica Location Service (RLS)* [18]
`file`	local to the host file name with a full path

An URL can be used in a standard form, i.e.

```
<protocol>://host[:port]/<file>
```

Or, to enhance the performance, it can have additional options:

```
<protocol>://host[:port][;option[;option[...]]]/<file>
```

For a metadata service URL, construction is the following:

```
lfc://[url[|url[...]]@]<host>[:port]/<lfn> rls://[url[|url[...]]@]<host>[:port]/<lfn>
rc://rc://[location[|location[...]]@]<host>[:port]/<DN>/<lfn>
fireman://[url[|url[...]]@]<host>[:port]/<service_path>?<lfn>
```

For the Smart Storage Element service, the syntax is

```
se://host[:port][;options]/path[?file_id]
```

For the SRM service, the syntax is

```
srm://<host>[:port][;options]/[service_path?SFN=]<file_id>
```

Versions 1.1 and 2.2 of the SRM protocol are supported. The default *service_path* is srm/managerv2 when the server supports v2.2, srm/managerv1 otherwise.

The URL components are:

`location`	`<location_name_in_RC>[;option[;option[...]]]`
`host[:port]`	IP address of a server
`DN`	Distinguished Name (as in LDAP) of an RC collection
`lfn`	Logical File Name (LFN)
`url`	URL of the file as registered in LFC/RLS/Fireman
`service_path`	End-point path of the Web service
`file`	local to the host file name with a full path

The following options are supported for location URLs:

`threads=<number>`	specifies number of parallel streams to be used by GridFTP or HTTP(s,g); default value is 1, maximal value is 10
`cache=yes|no|renew|copy`	indicates whether the GM should cache the file; default for input files is `yes`. `renew` forces a download of the file, even if the cached copy is still valid. `copy` forces the cached file to be copied (rather than linked) to the session dir, this is useful if for example the file is to be modified (`copy` option is available in ARC 0.8.1 and above).
`readonly=yes|no`	for transfers to `file://` destinations, specifies whether the file should be read-only (unmodifiable) or not; default is `yes`
`secure=yes|no`	indicates whether the GridFTP data channel should be encrypted; default is `no`
`blocksize=<number>`	specifies size of chunks/blocks/buffers used in GridFTP or HTTP(s,g) transactions; default is protocol dependent
`checksum=cksum|md5|adler32|no`	specifies the algorithm for checksum to be computed (for transfer verification or provided to the indexing server). This is overridden by any metadata options specified (see below). If this option is not provided, the default for the protocol is used. `checksum=no` disables checksum calculation (`adler32` and `no` options are available in ARC 0.8.1 and above).
`exec=yes|no`	means the file should be treated as executable
`preserve=yes|no`	specify if file must be uploaded to this destination even if job processing failed (default is `no`)
`pattern=<pattern>`	defines file matching pattern; currently works for file listing requests sent to an `se://` endpoint
`guid=yes|no`	make software use GUIDs instead of LFNs while communicating to indexing services; meaningful for `rls://` only
`overwrite=yes|no`	make software try to overwrite existing file(s), i.e. before writing to destination, tools will try to remove any information/content associated with specified URL
`protocol=gsi|gssapi`	to distinguish between two kinds of `httpg`. `gssapi` stands for implemention using only GSSAPI functions to wrap data and `gsi` uses additional headers as implmented in Globus IO
`spacetoken=<pattern>`	specify the space token to be used for uploads to SRM storage elements supporting SRM version 2.2 or higher
`autodir=yes|no`	specify if before writing to specified location software should try to create all directories mentioned in specified URL. Currently this applies to FTP and GridFTP only. Default for those protocols is `yes`

Local files are referred to by specifying either a location relative to the job submission working directory, or by an absolute path (the one that starts with "/"), preceded with a `file://` prefix.

Metadata service URLs also support metadata options which can be used for register additional metadata attributes or query the service using metadata attributes. These options are specified at the end of the LFN and consist of name and value pairs separated by colons. The following attributes are supported:

`guid`	GUID of the file in the metadata service
`checksumtype`	Type of checksum. Supported values are cksum (default), md5 and ad (adler32 checksum)
`checksumvalue`	The checksum of the file

Currently these metadata options are only supported for lfc:// URLs.

Examples of URLs are:

```
http://grid.domain.org/dir/script.sh
gsiftp://grid.domain.org:2811;threads=10;secure=yes/dir/input_12378.dat
ldap://grid.domain.org:389/lc=collection1,rc=Nordugrid,dc=nordugrid,dc=org
rc://grid.domain.org/lc=collection1,rc=Nordugrid,dc=nordugrid,dc=org/zebra/f1.zebra
rls://gsiftp://se.domain.org/datapath/file25.dat@grid.domain.org:61238/myfile02.dat[1]
fireman://fireman_host:8443/glite-data-catalog-interface/FiremanCatalog?data.root
file:///home/auser/griddir/steer.cra
lfc://srm://srm.domain.org/griddir@lfc.domain.org//user/file1:guid=\
    bc68cdd0-bf94-41ce-ab5a-06a1512764dc:checksumtype=ad:checksumvalue=12345678[2]
lfc://;cache=no@lfc.domain.org/:guid=bc68cdd0-bf94-41ce-ab5a-06a1512764d[3]
```

[1]This is a destination URL. The file will be copied to the GridFTP server at `se.domain.org` with the path `datapath/file25.dat` and registered in the RLS indexing service at `grid.domain.org` with the LFN `myfile02.dat`.

[2]This is a destination URL. The file will be copied to `srm.domain.org` at the path `griddir/file1` and registered to the LFC service at `lfc.domain.org` with the LFN `/user/file1`. The given GUID and checksum attributes will be registered.

[3]This is a source URL. The file is registered in the LFC service at `lfc.domain.org` with the given GUID and can be copied or queried by this URL. Note that as URL options are part of the location (physical) URL, in meta service URLs the options must be part of the location URL, even if the location URL is empty.

7.1.2 Task Description: Attributes

A task is typically described by a binary executable to be processed by a computer, parameters passed to that executable, and requirements which must be met by a system to be able to execute the task. For example, a simple "Hello World" task description would specify `/bin/echo` as the executable and "Hello World" as the argument. In xRSL language it will look as follows:

```
&(executable="/bin/echo")(arguments="Hello World")
```

You can actually use this description from the command line, i.e., your first Grid job should be submitted as*:

```
ngsub -e '&(executable="/bin/echo")(arguments="Hello World")'
```

In a more complex case, a binary executable may have to be downloaded from a remote location, process an input data and create an output file. This task specification will have to include locations of all three files: executable, input and output, and perhaps few other requirements, e.g., the necessary disk space and an e-mail address to which the notification of the task completion should be sent. Such an xRSL file may be written like the following:

```
&
(executable="myprog")
(inputFiles=
   ("myprog" "http://www.myserver.org/myfiles/myprog")
   ("myinput" "gsiftp://www.mystorage.org/data/file007.dat")
)
(outputFiles=
   ("myoutput" "gsiftp://www.mystorage.org/results/file007.res")
)
(disk=1000)
(notify="e myname@mydomain.org")
```

*When using ARC $\leq$ 0.5.48 client, omit `-e`

As you can see, the xRSL attribute names are largely self-explanatory. xRSL handles most possible task description attributes, and next Sections contain the comprehensive list of them, complete with usage examples.

The following attributes can be specified in a user's xRSL script. Some are to be modified by the UI before being passed to the GM. If this is the case, corresponding GM input is described.

executable

(ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(executable=<string>)`
GM input:	for ARC $\leq$ 0.5.25: `(executable="/bin/echo")`
Example:	`(executable="local_to_job.exe")`

The executable to be submitted as a main task to a Local Resource Management System (LRMS).

`string` file name (including path), local to the computing element (CE)

Executable is a file that has to be executed as the main process of the task. It could be either a pre-compiled binary, or a script. Users may transfer their own executables, or use the ones known to be already installed on the remote system (CE).

If an executable has to be transferred to the destination site (CE) from some source, it has to be specified in the `inputFiles` list. If it is not specified in `inputFiles`, the source is expected to be local to the user (client) and will be added as such to the `inputFiles` list by the ARC Client.

If the file name starts with a leading slash ("`/`"), it is considered to be **the full path to the executable at the destination site (CE)**; otherwise the location of the file is **relative** to the session directory (where job input and files are stored).

If the xRSL string is entered from the command line and is enclosed in double quotes, standard shell expansion of variables takes place. That is, if the file name contains an environment variable ("`$...`"), the value of this variable is resolved locally, but if the name itself is also enclosed in double quotes, it will be resolved at the remote computing element:
`(executable=$ROOT_DIR/myprog.exe)` – `$ROOT_DIR` is resolved locally (*will cause errors if the path does not exist at the execution machine*)
`(executable="$ROOT_DIR/myprog.exe")` – `$ROOT_DIR` will be resolved remotely

arguments

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(arguments=<string> [string] ... )`
GM input:	`(arguments=<executable> <string> [string] ... )`
Example:	`(arguments="10000" $(ATLAS)/input.dat)`

List of the arguments for the executable.

`string`	an argument
`executable`	the executable to be run by LRMS, taken by the ARC Client from the user-specified `executable` attribute

inputFiles

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: yes

Operators: `=`

User input: `(inputFiles=(<filename> <source>) ... )`

GM input:
```
(inputFiles=(<filename> <URL>)
        (<filename> [size][.checksum]) ... )
```

Example:
```
(inputFiles=("local_to_job" "gsiftp://se1.lu.se/p1/remote_one")
        ("local_to_job.dat" "/scratch/local_to_me.dat")
        ("same_name_as_in_my_current_dir" ""))
```

List of files to be copied to the computing element before the execution.

`filename`	destination file name, local to the computing element and always relative to the session directory
`source`	source of the file: (`gsiftp`, `https`, `ftp`, `http` URLs, or a path, local to the submission node). If void (””, use the quotes!), the input file is taken from the submission directory.
`URL`	URL of the file (see Section 7.1.1)
`size`	file size in bytes
`checksum`	file checksum (as returned by `cksum`)

ARC Client does not check whether the same destination name is specified for different source locations. Make sure all `filename` values are different!

If the list does not contain the standard input file (as specified by `stdin`) and/or the executable file (as specified by `executable` if the given name), the ARC Client appends these files to the list. If the `<source>` is a URL, it is passed by the ARC Client to the GM without changes; if it is a local path (or void, ””), the ARC Client converts it to `<size>` and `<checksum>` and uploads those files to the execution cluster.

Please note that the `inputFiles` attribute is not meant to operate with directories, for reasons of access control and checksum verifications. You must specify a pair `("<local_to_job>" "<source>")` for each file.

executables

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: yes

Operators: `=`

User input: `(executables=<string> [string] ...)`

GM input: `-"-`

Example: `(executables="myscript.sh" "myjob.exe")`

List of files from the `inputFiles` set, which will be given executable permissions.

`string`	file name, local to the computing element and relative to the session directory

If the executable file (as specified in `executable` and is relative to the session directory) is not listed, it will be added to the list by the ARC Client.

cache

(ARC $\geq$ 0.3.34, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(cache="yes"|"no")`
GM input:	`-"-`
Example:	`(cache="yes")`

Specifies whether input files specified in the `inputFiles` should be placed by default in the cache or not. This does not affect files described by the `executables`, which will be placed in the session directory always.

If not specified, default value is "yes".

> Cached files can not be modified by jobs by default. If your job has to modify input files, please use the `(readonly="no")` URL option for those files. This option does not affect whether or not the file is cached.

outputFiles

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(outputFiles=(<string> <URL>) ... )`
GM input:	`-"-`

Example:	`(outputFiles=("local_to_job.dat" "gsiftp://se1.uo.no/stored.dat")`
	`("local_to_job.dat" "rc://rc1.uc.dk/set1/stored_and_indexed.dat")`
	`("local_to_job_dir/" ""))`

List of files to be retrieved by the user or uploaded by the GM and indexed (registered) in a data indexing service, e.g. Globus® RLS or Fireman.

`string`	file name, local to the *Computing Element (CE)*. If this string ends with a backslash "/" and `<URL>` is empty, the entire directory is being kept at the execution site. If however this string ends with a backslash "/" but the `<URL>` starts with `gsiftp` or `ftp`, the whole directory is transfered to the destination.
`URL`	destination URL of the remote file (see Section 7.1.1); if void (""", use the quotes!), the file is kept for manual retrieval. Note that this can not be a local `file://` URL.

Using a Replica Catalog pseudo-URL (see Section 7.1.1), you should make sure that the location is already defined in the Replica Catalog. If more than one such location is specified, only those found in the Replica Catalog for this collection will be used. Grid Manager will store output files in **one** location only: the first randomly picked location that exists. If no locations are specified, all those found in the Replica Catalog for this collection will be tried.

If in a `rc://` pseudo-URL the server component `host[:port]/DN` is not specified, server specified in the `replicaCollection` attribute will be used.

If the list does not contain standard output, standard error file names and GM log-files directory name (as specified by `stdout`, `stderr` and `gmlog`), the ARC Client appends these items to the `outputFiles` list. If the `<URL>` is not specified (void, """, use the quotes!), files will be kept on SE and should be downloaded by the user via the ARC Client. If specified name of file ends with "/", the entire directory is kept.

A convenient way to keep the entire job directory at the remote site for a manual retrieval is to specify `(outputfiles=("/" ""))`.

In some cases, the list of output files may only be known after the job has completed. ARC allows a user to specify a list of output files dynamically in a file or files in the session directory as part of their job. The file(s) containing the output file information can be defined in the xRSL script as the path to the file relative to the session directory preceeded by '@'. The format of these files is lines of 2 values separated by a space. The first value contains name of the output file relative to the session directory and the second value is a URL to which the file will be uploaded.

Example: `(outputFiles=("@output.files" "")`

output.files is generated by the user and contains

```
file1 gsiftp://grid.domain.org/file1
file2 gsiftp://grid.domain.org/file2
```

After the job completes, the file output.files in the session directory will be read and any files described within will be uploaded to the given URLs.

> Please note that the `outputFiles` attribute is not meant to operate with directories: you must specify a pair `("<local_to_job>" "[destination]")` for each file. One exception is when you want to preserve an entire directory at the remote computer for later **manual** download via `ngget`. In that case, simply add the trailing backslash "/" at the end of the remote directory name. You can not upload a directory to a URL location, only to your local disk.

cpuTime

(ARC $\geq$ 0.3.17, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(cpuTime=<time>)`
GM input:	`(cpuTime=<tttt>)`
Example:	`(cpuTime="240")`

Maximal CPU time request for the job. For a multi-processor job, this is a sum over all requested processors.

`time`	time (in minutes if no unit is specified)
`tttt`	time converted by the ARC Client from `time` to seconds.

The client converts time specified in the user-side XRSL file to seconds. If no time unit is specified, the client assumes the time given in minutes. Otherwise, a text format is accepted, i.e., any of the following will be interpreted properly (make sure to enclose such strings in quotes!):

```
"1 week"
"3 days"
"2 days, 12 hours"
"1 hour, 30 minutes"
"36 hours"
"9 days"
"240 minutes"
```

If both `cpuTime` and `wallTime` are specified, the ARC Client converts them both. `cpuTime` can not be specified together with `gridTime` or `benchmarks`.

This attribute should be used to direct the job to a system with sufficient CPU resources, typically, a batch queue with the sufficient upper time limit. Jobs exceeding this maximum most likely will be **terminated** by remote systems! If time limits are not specified, the limit is not set and jobs can run as long as the system settings allow (note that in this case you can not avoid queues with too short time limits).

wallTime

(ARC $\geq$ 0.5.27, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(wallTime=<time>)`
GM input:	`(wallTime=<tttt>)`
Example:	`(wallTime="240")`

Maximal wall clock time request for the job.

`time`	time (in minutes if no unit is specified)
`tttt`	time converted by the ARC Client to seconds

The client converts time specified in the user-side XRSL file seconds. If no time unit is specified, the client assumes the time given in minutes. Otherwise, a text format is accepted, i.e., any of the following will be interpreted properly (make sure to enclose such strings in quotes!):

```
"1 week"
"3 days"
"2 days, 12 hours"
"1 hour, 30 minutes"
"36 hours"
"9 days"
"240 minutes"
```

If both `cpuTime` and `wallTime` are specified, the ARC Client converts them both. `wallTime` can not be specified together with `gridTime` or `benchmarks`. If only `wallTime` is specified, but not `cpuTime`, the corresponding `cpuTime` value is evaluated by the ARC Client and added to the job description.

This attribute should be used to direct the job to a system with sufficient CPU resources, typically, a batch queue with the sufficient upper time limit. Jobs exceeding this maximum most likely will be **terminated** by remote systems! If time limits are not specified, the limit is not set and jobs can run as long as the system settings allow (note that in this case you can not avoid queues with too short time limits).

gridTime

(ARC $\geq$ 0.3.31, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(gridTime=<time>)`
GM input:	none
Example:	`(gridTime="2 h")`

Maximal CPU time request for the job scaled to the 2.8 GHz Intel® Pentium® 4 processor.

`time` time (in minutes if no unit is specified)

The attribute is completely analogous to `cpuTime`, except that it will be recalculated to the actual CPU time request for each queue, depending on the published processor clock speed.

`gridTime` can not be specified together with `cpuTime` or `wallTime`. If only `gridTime` is specified, but not `cpuTime`, the corresponding `cpuTime` value is evaluated by the ARC Client and added to the job description.

benchmarks

(ARC ≥ 0.5.14, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(benchmarks=(<string> <value> <time>) ... )`
GM input:	
Example:	`(benchmarks=("mybenchmark" "10" "1 hour, 30 minutes"))`

Evaluate a job's `cpuTime` based on benchmark values.

`string` benchmark name

`value` benchmark value of reference machine

`time` the `cpuTime` the job requires on the reference machine

`benchmarks` can not be specified together with `cpuTime` or `wallTime`. If only `benchmarks` is specified, but not `cpuTime`, the corresponding `cpuTime` value is evaluated by the ARC Client and added to the job description.

memory

(ARC ≥ 0.3.17†, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(memory=<integer>)`
GM input:	`-"-`
Example:	`(memory>="500")`

Memory required for the job, per rank.

`integer` size (Mbytes)

> Similarly to `cpuTime`, this attribute should be used to direct a job to a resource with a sufficient capacity. Jobs exceeding this memory limit will most likely be **terminated** by the remote system.

disk

(ARC ≥ 0.3.17‡, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

†Was called `maxMemory` for ARC ≤ 0.3.16

‡Was called `maxDisk` for ARC ≤ 0.3.16

Unique: no

Operators: `=` `!=` `>` `<` `>=` `<=`

User input: `(disk=<integer>)`

GM input: none

Example: `(disk="500")`

Disk space required for the job.

`integer` disk space, Mbytes

This attribute is used at the job submission time to find a system with sufficient disk space. However, it **does not guarantee** that this space will be available at the end of the job, as most known systems do not allow for disk space allocation. Eventually, a remote system can terminate a job that exceeds the requested disk space.

runTimeEnvironment

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: no

Operators: `=` `!=` `>` `<` `>=` `<=`

User input: `(runTimeEnvironment=<string>)(runTimeEnvironment=<string>)`

GM input: For ARC $\leq$ 0.5.39: `(runTimeEnvironment="<string>" "<string>")`

Example: `(runTimeEnvironment>="APPS/HEP/ATLAS-10.0.1")`

Required runtime environment.

`string` environment name

The site to submit the job to will be chosen by the ARC Client among those advertising specified runtime environments. Before starting the job, the GM will set up environment variables and paths according to those requested. Runtime environment names are defined by Virtual Organizations, and tend to be organized in name spaces.

To request several environments, repeat the attribute string:
`(runTimeEnvironment="ENV1")(runTimeEnvironment="ENV2")` etc.

To make a disjunct-request, use a boolean expression:
`(|(runTimeEnvironment="env1")(runTimeEnvironment="env2"))`.

You can use "`>=`" or "`<=`" operators: job will be submitted to any suitable site that satisfies such requirements, and among the available at the sites runtime environments, the highest version satisfying a requirement will be requested in the pre-processed xRSL script.

Runtime environment string interpretation is case-insensitive. If a runtime environment string consists of a name and a version number, a partial specification is possible: it is sufficient to request only the name and use "`>`" or "`>=`" operators to select the highest version.

middleware

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: no

Operators: `=` `!=` `>` `<` `>=` `<=`

User input: `(middleware=<string>)`

GM input: `-"-`

Example: `(middleware="nordugrid-arc-0.5.99")`

Required middleware version. Make sure to specify full name and version number.

`string` Grid middleware name.

The site to submit the job to will be chosen by the ARC Client among those advertising specified middleware. Usage is identical to that of the `runTimeEnvironment`. Use the ">=" operator to request a version "equal or higher".

opsys

(ARC ≥ 0.3.21, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	no
Operators:	`=` `!=` `>` `<` `>=` `<=`
User input:	`(opsys=<string>)`
GM input:	`-"-`
Example:	`(opsys="FC3")`

Required operating system.

`string` Operating system name and version.

The site to submit the job to will be chosen by the ARC Client among those advertising specified operating system. Usage is identical to that of `runTimeEnvironment` and `middleware`. Use the ">=" operator to request a version "equal or higher".

stdin

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(stdin=<string>)`
GM input:	`-"-`
Example:	`(stdin="myinput.dat")`

The standard input file.

`string` file name, local to the computing element

The standard input file should be listed in the `inputFiles` attribute; otherwise it will be forced to that list by the ARC Client.

stdout

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(stdout=<string>)`
GM input:	`-"-`
Example:	`(stdout="myoutput.txt")`

The standard output file.

`string` file name, local to the computing element and relative to the session directory.

The standard output file should be listed in the `outputFiles` attribute; otherwise it will be forced to that list by the ARC Client. If the standard output is not defined, ARC Client assigns a name.

stderr

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(stderr=<string>)`
GM input:	`-"-`
Example:	`(stderr="myjob.err")`

The standard error file.

`string`	file name, local to the computing element and relative to the session directory.

The standard error file should be listed as an `outputFiles` attribute; otherwise it will be forced to that list by the ARC Client. If the standard error is not defined, ARC Client assigns a name. If `join` is specified with value "yes", ARC Client adds `stderr` to the pre-processed xRSL script with the same value as `stdout`.

join

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(join="yes"|"no")`
GM input:	none
Example:	`(join="yes")`

If "yes", joins `stderr` and `stdout` files into the `stdout` one. Default is `no`.

gmlog

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(gmlog=<string>)`
GM input:	`-"-`
Example:	`(gmlog="myjob.log")`

A name of the directory containing grid-specific diagnostics per job.

`string`	a directory, local to the computing element and relative to the session directory

This directory is kept in the session directory to be available for retrieval (ARC Client forces it to the list if `outputFiles`)

jobName

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(jobName=<string>)`
GM input:	`-"-`
Example:	`(jobName="My Job nr. 1")`

User-specified job name.

`string` job name

This name is meant for convenience of the user. It can be used to select the job while using the ARC Client. It is also available through the Information System.

ftpThreads

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(ftpThreads=<integer>)`
GM input:	`-"-`
Example:	`(ftpThreads="4")`

Defines how many parallel streams will be used by the GM during `gsiftp` and `http(s|g)`transfers of files.

`integer` a number from 1 to 10

If not specified, parallelism is not used.

acl

(ARC $\geq$ 0.5.12, ARC 0.6, ARC 0.8)

Unique:	no
Operators:	`=`
User input:	`(acl=<xml>)`
GM input:	`-"-`
Example:	`(acl="<?xml version=""1.0""?>`
	`<gacl version=""0.0.1""><entry><any-user></any-user>`
	`<allow><write/><read/><list/><admin/></allow></entry></gacl>")`

Makes use of GACL [19] rules to list users who are allowed to access and control job in addition to job's owner. Access and control levels are specified per user. `any-user` tag refers to any user authorized at the execution cluster. To get more information about GACL please refer to `http://www.gridsite.org`.

`xml` a GACL-compliant XML string defining access control list

Following job control levels can be specified via `acl`:

- `write` – allows to modify contents of job data (job directory) and control job flow (cancel, clean, etc.)
- `read` – allows to read content of job data (contents of job directory)
- `list` – allows to list files available for the job (contents of job directory)
- `admin` – allows to do everything – full equivalence to job ownership

cluster

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=` `!=`
User input:	`(cluster=<string>)`
GM input:	`-"-`
Example:	`(cluster="cluster1.site2.org")`

The name of the execution cluster.

`string` known cluster name, or a substring of it

Use this attribute to explicitly force job submission to a cluster, or to avoid such. The job will not be submitted if the cluster does not satisfy other requirements of the job. Disjunct-requests of the kind `(|(cluster="clus1")(cluster="clus2"))` are supported. To exclude a cluster, use `(cluster!="clus3")`.

queue

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=` `!=`
User input:	`(queue=<string>)`
GM input:	`-"-`
Example:	`(queue="pclong")`

The name of the remote batch queue.

> Use only when you are sure that the queue by this name does exist.

`string` known queue name

While users are not expected to specify `queue` in job descriptions, this attribute **must** be present in the GM-side xRSL. In fact, this is primarily an internal attribute, added to the job description by client tools after resource discovery and matchmaking. Still, users can specify this attribute to explicitly force job submission to a queue: when specified explicitly by the user, this value will not be overwritten by the ARC Client, and an attempt will be made to submit the job to the specified queue.

If for some reason (e.g. due to a client tool error) `queue` is absent from the GM-side xRSL, GM on the selected cluster will attempt to submit the job to the default queue if such is specified in the GM configuration.

startTime

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(startTime=<time>)`
GM input:	`(startTime=<tttt>)`
Example:	`(startTime="2002-05-25 21:30")`

Time to start job processing by the Grid Manager, such as e.g. start downloading input files.

`time` time string, YYYY-MM-DD hh:mm:ss

`tttt` time string, YYYYMMDDhhmmss[Z] (converted by the ARC Client from `time`)

Actual job processing on a worker node starts depending on local scheduling mechanisms, but not sooner than `startTime`.

lifeTime

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(lifeTime=<time>)`
GM input:	`(lifeTime=<tttt>)`
Example:	`(lifeTime="2 weeks")`

Maximal time to keep job files (the session directory) on the gatekeeper upon job completion.

`time` time (in minutes if no unit is specified)

`tttt` time (seconds, converted by the ARC Client from `time`)

Typical life time is 1 day (24 hours). Specified life time can not exceed local settings.

notify

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(notify=<string> [string] ... )`
GM input:	`-"-`
Example:	`(notify="be your.name@your.domain.com")`

Request e-mail notifications on job status change.

`string` string of the format: `[b][q][f][e][c][d] user1@domain1 [user2@domain2] ...`
here flags indicating the job status are:
`b` – begin (PREPARING)
`q` – queued (INLRMS)
`f` – finalizing (FINISHING)
`e` – end (FINISHED)
`c` – cancellation (CANCELLED)
`d` – deleted (DELETED)

When no notification flags are specified, default value of "`eb`" will be used, i.e., notifications will be sent at the job's beginning and at its end.

No more than 3 e-mail addresses per status change accepted.

replicaCollection

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: no
Operators: `=`
User input: `(replicaCollection=<URL>)`
GM input: `-"-`
Example: `(replicaCollection="ldap://grid.uio.no:389/lc=TestCollection, rc=NorduGrid,nordugrid,dc=org")`

Location of a logical collection in the Replica Catalog.

`URL` LDAP directory specified as an URL (`ldap://host[:port]/dn`)

rerun

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: yes
Operators: `=`
User input: `(rerun=<integer>)`
GM input: `-"-`
Example: `(rerun="2")`

Number of reruns (if a system failure occurs).

`integer` an integer number

If not specified, the default is 0. Default maximal allowed value is 5. The job may be rerun after failure in any state for which reruning has sense. To initiate rerun user has to use the `ngresume` command.

architecture

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: no
Operators: `=` `!=`
User input: `(architecture=<string>)`
GM input:
Example: `(architecture="i686")`

Request a specific architecture.

`string` architecture (e.g., as produced by uname -a)

nodeAccess

(ARC $\geq$ 0.3.24, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: yes
Operators: `=`
User input: `(nodeAccess="inbound"|"outbound")`
GM input:
Example: `(nodeAccess="inbound")`

Request cluster nodes with inbound or outbound IP connectivity. If both are needed, a conjunct request should be specified.

dryRun

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	yes
Operators:	`=`
User input:	`(dryRun="yes"|"no")`
GM input:	`-"-`
Example:	`(dryRun="yes")`

If "yes", do dry-run: job description is sent to the optimal destination, input files are transferred, but no actual job submission to LRMS is made. Typically used for xRSL and communication validation.

rsl_substitution

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	no
Operators:	`=`
User input:	`(rsl_substitution=(<string1> <string2>))`
GM input:	`-"-`
Example:	`(rsl_substitution=("ATLAS" "/opt/atlas"))`

Substitutes `<string2>` with `<string1>` for **internal** RSL use.

`string1`	new internal RSL variable
`string2`	any string, e.g., existing combination of variables or a path

Use this attribute to define variables that simplify xRSL editing, e.g. when same path is used in several values, typically in `inputFiles`. Only one pair per substitution is allowed. To request several substitution, concatenate such requests. Bear in mind that substitution must be defined **prior** to actual use of a new variable `string1`.

After the substitution is defined, it should be used in a way similar to shell variables in scripts: enclosed in round brackets, preceded with a dollar sign, **without quotes**:
`(inputfiles=("myfile" $(ATLAS)/data/somefile))`

Unlike the `environment` attribute, `rsl_substitution` definition is only used by the client and is valid inside xRSL script. It can not be used to define environment or shell variable at the execution site.

environment

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique:	no
Operators:	`=`
User input:	`(environment=(<VAR> <string>) [(<VAR> <string>)] ... )`
GM input:	`-"-`
Example:	`(environment=("ATLSRC" "/opt/atlas/src")` `("ALISRC" "/opt/alice/src"))`

Defines execution shell environment variables.

`VAR`	new variable name
`string`	any string, e.g., existing combination of variables or a path

Use this to define variables at an execution site. Unlike the `rsl_substitution` attribute, it can not be used to define variables on the client side.

count

(ARC ≥ 0.3.12, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: yes
Operators: =
User input: (count=<integer>)
GM input: -"-
Example: (count="4")

Specifies amount of sub-jobs to be submitted for parallel tasks.

jobreport

(ARC ≥ 0.3.34, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Unique: yes
Operators: =
User input: (jobreport=<URL>)
GM input: -"-
Example: (jobreport="https://grid.uio.no:8001/logger")

Specifies an URL for a logging service to send reports about job to. The default is set up in the cluster configuration.

URL URL

It is up to a user to make sure the requested logging service accepts reports from the set of clusters she intends to use.

credentialserver

(ARC ≥ 0.5.45, ARC 0.6, ARC 0.8)

Unique: yes
Operators: =
User input: (credentialserver=<URL>)
GM input: -"-
Example: (credentialserver="myproxy://myproxy.nordugrid.org;username=user")

Specifies an URL which Grid Manager may contact to renew/extend delegated proxy of job. Only MyProxy servers are supported.

URL URL of MyProxy server

It is up to a user to make sure the specified MyProxy server will accept requests from Grid Manager to renew expired credentials. URL may contain options username and credname to specify user name and credentials name which Grid Manager should pass to MyProxy server. If username is not specified DN of user credentials is used instead.

7.2 Examples

In this section you can find some examples of xRSL job description.

While the xRSL specifications can be passed to ngsub (see Section 8.2.2) as a single string on the command line, it is much more convenient to create a file with an arbitrary name, containing the xRSL string, and

pass this file to ngsub, e.g. `ngsub filename`. The format of the file is not important (see Section 7.1), as long as it does not contain Microsoft/DOS linefeeds.

Attribute names are case-insensitive, that is, you can use e.g. `CPUTime` or `cputime` interchangeably. The only attribute which **must** be specified is `executable`, others can be added by the user depending on the needs, in no particular order.

7.2.1 Hello World

The simplest way to say "Hello World" is to assume that any cluster on the Grid has the Unix `echo` executable in the `/bin/` directory. In most cases it will work, so the following xRSL file can be composed:

```
&
(* main executable of the task *)
 (executable=/bin/echo)
(* arguments for the main executable *)
 (arguments="Hello World" )
(* standard output will be redirected to this file: *)
 (stdout="hello.txt")
(* standard error will be redirected to this file: *)
 (stderr="hello.err")
(* Grid Manager auxilliary logs will be stored in this directory: *)
 (gmlog="gridlog")
(* give job a distinct name for easy monitoring *)
 (jobname="My Hello Grid")
(* instruct cluster that your job should be placed in a queue with *)
(* the sufficient time limit for the job to get completed *)
 (cputime=5)
(* chose only those clusters which have a proper ARC installation *)
 (middleware>="nordugrid-arc-0.3.24")
```

In this example, strictly speaking, only the `executable` and `arguments` are needed to describe the task. However, as the job will be executed in batch (like everything on the Grid), you will never see the message "Hello World" on your screen. Instead, you should instruct the Grid Manager to capture the standard output to a file, which you can later retrieve. This is how `stdout` attribute appears in the xRSL. It may happen that something goes wrong (e.g., `echo` is located in `/usr/bin/`), and an error will be produced. To be able to analyze the errors, users are advised to use `stderr` and `gmlog` attributes, which instruct the Grid Manager where to store possible error messages.

The job name is a convenient way to identify the task. Although every job is assigned a unique Grid ID, this ID is far from intuitive. As soon as you plan to submit more than one job, you should think of a set of good names for them – this will save you a lot of time. The attribute `jobname` should be used to associate every job with a user-defined name.

Since the Grid will parse the task to a batch system, it is always a good idea to specify the needed CPU time. Some clusters are known to have the default execution time set to 0, which effectively kills any job. The `cputime` attribute takes the time value (in minutes) and passes it to the Grid Manager. Do not overestimate the time, as your job may end up in a "long" queue, and wait an unnecessarily long time before being executed.

The last attribute in this example is `middleware`. This may be quite helpful in an environment where some clusters run untested versions of the middleware. When you are not sure what to specify, do `ngsub -v`, and use the printed version number.

7.2.2 An Own Executable

In the previous Section, the `executable` attribute had the value `/bin/echo`.

> A very important thing to understand is that the Grid Manager always expects the file specified by the `executable` attribute to be at the execution machine, not the user's machine.

This means that if a user wants to submit a job executing an own tool `say_hello` and specifies in the xRSL text `(executable=say_hello)`, the Grid Manager will attempt to execute a file `say_hello` in the temporary session directory, local to the job. Naturally, this file will not appear there by magic. Instead, the User Interface, prior to job submission, will interpret such a request as if a user has the executable file `say_hello` in the current directory at the submission machine, and will upload the file to the destination. Absence of the leading slash (`/`) makes the User Interface assume that the file in question resides locally in the path relative to the current directory.

So far, so good, but what if the executable resides not in the current directory, but elsewhere on your computer, or, even better, at a remote location? If this is the case, you have to instruct the User Interface how to get the file by using the `inputfiles` attribute:

```
&
(* main executable of the task *)
 (executable=say_hello)
(* arguments for the main executable *)
 (arguments="Hello again" )
(* where does the executable reside *)
 (inputfiles=(say_hello file:///home/john/bin/say_hello))
(* standard output will be redirected to this file: *)
 (stdout="hello.txt")
```

The `inputfiles` attribute instructs the User Interface to copy the file from `/home/john/bin/say_hello` on your local computer to `say_hello` in the session directory at the execution machine. If the file resides at an GridFTP or HTTP server, use `gsiftp://` or `http://` respectively instead of `file://`to specify the protocol.

7.2.3 Many Executables

The Grid Manager is capable of managing more than one executable file, and xRSL provides means for users to specify such files. A typical example is when a user script prepares and starts a pre-compiled binary. In this case, both files must be given executable permissions, but only the "main" script should be submitted for execution by the local system. To make this working, users must list all the files which need executable permissions in the `executables` attribute, as shown below:

```
&
(executable=run.sh)
(arguments=1664900 100000)
(executables=ffungen)
(inputFiles=(ffungen ""))
(outputFiles=(ffun.hbook gsiftp://hathi.hep.lu.se/test/flong/flong1.hbook))
(jobName=flong1)
(stdout=flong1.out)
(join=yes)
```

```
(notify="e john.doe@mail.org")
(ftpThreads=6)
(middleware="NorduGrid-0.8.1")
```

Chapter 8

Job Management

Job submission on the NorduGrid/ARC is made via the *User Interface (UI)* part of the ARC [5]. The UI provides a set of command line tools that submit jobs, trace their status, kill jobs, retrieve job output, and perform some other related functions. For a complete description and configuration details, please refer to the UI manual [5]. The following sections will concentrate on job and data management tools and utilities in the ARC client.

8.1 First steps: the test suite

ARC comes with a fairly complete test-suite that tests different functionalities of the middleware. The test-suite consists of one utility, called `ngtest`, that contains a variety of test examples which can be executed. This utility can be used either for testing a newly installed cluster or by a new user that wants to try out a few simple things on the Grid for the first time. The utility can print helpful output about installed user credentials, user authorization on computing resources and storage elements, VO membership information, and it can also be used to submit various test-jobs, testing either the software installation on the client or some server.

The utility comes with the `nordugrid-arc-client` and the `nordugrid-arc-standalone` packages.

8.1.1 ngtest

ngtest [options]

(ARC $\geq$ 0.5.49*)

Options:

`-E, -certificate`		prints information about installed user- and CA-certificates
`-j, -job`	*number*	submit test job given by the number (1,2,3 possible)
`-R, -resources`		print user authorization information for clusters and storage elements
`-O, -configuration`		print user configuration
`-V, -vo`		print VO membership information for the user
`-r, -runtime`	*time*	run testjob 1 for the specified number of minutes (default 5)
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific cluster

*`ngtest` was available in ARC 0.4 and earlier 0.5.x versions, but had a substantially different functionality

`-C, -clustlist`	`[-]`textemfilename	list of clusters to select or reject
`-g, -giisurl`	*url*	URL of a central Information System server
`-G, -giislist`	*filename*	list of GIIS URLs
`-o, -joblist`	*filename*	file where the job IDs will be stored
`-dryrun`		add dryrun option to the xRSL
`-dumpxrsl`		do not submit – dump transformed xRSL to stdout
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Default values of some options (e.g. `timeout` or `giislist`) are shared with other client commands and can be specified in the user configuration file (Section 3.3).

Usage examples:

- discover resources available for the user:
  `ngtest -R`
- print out certificate information:
  `ngtest -E`
- submit a test job:
  `ngtest -J 1`
- submit a test job to a selected site:
  `ngtest -J 1 -c grid.place.org`
- submit a test job with dianostics printout :
  `ngtest -J 1 -d 2`

Test job descriptions:

1. This test-job calculates prime-numbers for 2 minutes and saves the list. The source-code for the prime-number program, the Makefile and the executable is downloaded to the cluster chosen from various servers and the program is compiled before the run. In this way, the test-job constitute a fairly comprehensive test of the basic setup of a grid cluster.
2. This test-job does nothing but print `hello, grid` to stdout.
3. This test-job tests download capabilities of the chosen cluster by downloading 8 inputfiles over different protocols – HTTP, FTP, gsiFTP and RLS.

8.2 Working with jobs

8.2.1 ngsync

It is advised to start every grid session by running `ngsync`, especially when changing workstations. The reason is that your job submission history is cached on your machine, and if you are using ARC client installations on different machines, your local lists of submitted jobs will be different. To synchronise these lists with the information in the Information System, use the `ngsync` command.

ngsync [options]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-g, -giisurl`	*url*	URL of a central Information System server
`-G, -giislist`	*filename*	list of GIIS URLs
`-f, -force`		don't ask for confirmation
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

The ARC ARC client keeps a local list of jobs in the user's home directory (see section 8.2.11). If this file is lost, corrupt, or the user wants to recreate the file on a different workstation, the `ngsync` command will recreate this file from the information available in the Information System.

Since the information about a job in the system can be slightly out of date in case the user submitted or removed a job very recently, a warning is issued when this command is run. The `-f` option disables this warning.

> As the Information System has a certain latency, do not use `ngsync` immediately after submitting or killing a job, first wait a few minutes.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

8.2.2 ngsub

The `ngsub` command is the most essential one, as it is used for submitting jobs to the Grid resources. . `ngsub` matches user's job description to the information collected from the Grid, and the optimal site is being selected for job submission. The job description is then being forwarded to that site, in order to be submitted to the Local Resource Management System (LRMS), which can be, e.g., PBS or Condor or SGE etc.

ngsub [options] <task ...>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-g, -giisurl`	*url*	URL of a central Information System server (GIIS)
`-G, -giislist`	*filename*	list of GIIS URLs
`-e, -xrsl`	*filename*	string describing the job to be submitted
`-f, -file`	*filename*	file describing the job to be submitted

`-o, -joblist`	*filename*	file where the job IDs will be stored
`-dryrun`		add dryrun option to the job description
`-dumpxrsl`		do not submit – dump transformed job description to stdout
`-U, -unknownattr`		allow unknown attributes in job description – pass them through
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page
Arguments:		
`task ...`		strings or files describing the jobs to be submitted

Option `-e` was introduced in ARC $\geq$ 0.5.49:
– in ARC $\leq$ 0.5.48, strings describing the job should not be pre-pended with option `-e`
– in ARC $\geq$ 0.5.49, files describing the job don't have to be pre-pended with option `-f`

A simple *"Hello World"* job would look like[†]:

```
    ngsub -e '&(executable="/bin/echo")(arguments="Hello
World")(stdout="hello.txt")'
```

Use single quotes for the xRSL string and double quotes for attribute values.

Such a request would submit the task to any available site, as there are no specific requirements specified. The job will be executed in a batch mode, and the standard output will be written to a file `hello.txt` at the execution site. You will have to retrieve this file manually, using the `ngget` command.

If a job is successfully submitted, a **job identifier** (*job ID*) is printed to standard output.

The job ID uniquely identifies the job while it is being executed. A typical job ID looks like follows:

```
    gsiftp://site.it.uni.org:2812/jobs/10308913211503407485
```

You should use this as a handle to refer to the job when doing other job manipulations, such as querying job status (`ngstat`), killing it (`ngkill`), re-submitting (`ngresub`), or retrieving the result (`ngget`).

Every job ID is a valid URL for the job session directory. You can always use it to access the files related to the job, by using data management tools (see Chapter 9).

The job description in xRSL or JSDL format can be given either as an argument on the command line, as in the example above, or can be **read from a file**[‡]. Several jobs can be requested at the same time by giving more than one filename argument, or by repeating the `-f` or `-e` options. It is possible to mix `-e` and `-f` options in the same `ngsub` command.

[†]When using ARC $\leq$ 0.5.48 client, omit `-e`
[‡]When using ARC $\leq$ 0.5.48, one must prepend file name with `-f`

To **validate** your job description without actually submitting a job, use the `-dryrun` option: it will capture possible syntax or other errors, but will instruct the site not to submit the job for execution.

If the `-o` option is given, the job identifier is also written to a file with the specified filename. This file can later be used with the corresponding `-i` option of the other job manipulating ARC client commands.

```
ngsub -o my_jobid_list myjob.xrsl
ngkill -i my_jobid_list
```

The `-c` option can be used to **force** a job to be submitted to a particular site (cluster), or to reject submission to a site. The matching is done by string match to the site name (i.e. hostname) as defined in the Information System, or to an alias, as defined in user configuration file (see Section 3.3). The `-c` option can be repeated several times, for example:

```
ngsub -c grid.nbi.dk -c grid.tsl.uu.se myjob.xrsl
```

This will submit a job to either `grid.nbi.dk` or `grid.tsl.uu.se`. To submit a job to any site except `badsite.abc.org`, use `-` sign in front of the name:

```
ngsub -c -badsite.abc.org myjob.xrsl
```

For convenience, you may list the sites in a file, and use the `-C` option to refer to the whole list:

```
ngsub -C preferred_sites myjob.xrsl
```

If a cluster name or file name is preceded with a minus sign ("-"), this site (or the list) will be avoided during the submission. This gives a possibility to blacklist unwanted sites:

```
ngsub -C -blacklist myjob.xrsl
```

The `ngsub` command locates the available sites by querying the Information System. By default, a list of the ARC Information System **servers** (GIIS) is distributed with the middleware and is stored in `$NORDUGRID_LOCATION/etc/giislist`. A user is free to choose any other set of GIIS servers, either by listing them in the configuration file[§], or by specifying them via `-g` option:

```
ngsub -g ldap://hostname[:port]/basedn myjob.xrsl
```

If the port number is omitted, the default 2135 is used. Two different syntax for the base DN of the LDAP query are accepted, e.g. a NorduGrid top-level GIIS can be specified in either of the two following ways:

```
ldap://index1.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
ldap://index1.nordugrid.org:2135/mds-vo-name=NorduGrid,O=Grid
```

Several GIISes can be specified by repeating the `-g` option:

```
ngsub -g ldap://url1 -g ldap://url2 ...
```

You may prefer to store the list of GIIS servers in a file other than the default one, and use the `-G` option to instruct `ngsub` to contact those servers instead:

```
ngsub -G my_GIIS_list myjob.xrsl
```

Note that LDAP URLs containing "Mds-Vo-name=local" refer not to GIISes, but to individual sites, for example:
`ldap://grid.tsl.uu.se:2135/O=Grid/Mds-Vo-name=local`
This feature can be used to add "hidden" sites that do not register to any GIIS, or to list explicitly preferred submission sites.

[§]For ARC $\leq$ 0.5.48, use `$HOME/.nggiislist` file

If neither the `-giisurl` nor the `-giislist` option is given, a list of GIIS URLs is read from the first existing source in the following list:

1. `giis` options in `client` section in `$HOME/.arc/client.conf`
2. `giis` options in `common` section in `$HOME/.arc/client.conf`
3. `$HOME/.nggiislist` (one GIIS per line, ARC $\leq$ 0.5.48)
4. `giis` options in `client` section in `$ARC_LOCATION/etc/arc.conf`
5. `giis` options in `common` section in `$ARC_LOCATION/etc/arc.conf`
6. `$ARC_LOCATION/etc/giislist` (one GIIS per line)
7. `giis` options in `client` section in `/etc/arc.conf`
8. `giis` options in `common` section in `/etc/arc.conf`
9. `/etc/giislist` (one GIIS per line)

If you would like to get diagnostics of the process of resource discovery and requirements matching, a very useful option is `-d`. The following command:

```
ngsub -d 2 myjob.xrsl
```

will print out the steps taken by the ARC client to find the best cluster satisfying your job requirements. A default value for the debug level can be set by the first existing of:

1. `debug` option in `client` section in `$HOME/.arc/client.conf`
2. `debug` option in `common` section in `$HOME/.arc/client.conf`
3. `NGDEBUG` option in `$HOME/.ngrc` (ARC $\leq$ 0.5.48)
4. `debug` option in `client` section in `$ARC_LOCATION/etc/arc.conf`
5. `debug` option in `common` section in `$ARC_LOCATION/etc/arc.conf`
6. `debug` option in `client` section in `/etc/arc.conf`
7. `debug` option in `common` section in `/etc/arc.conf`

It often happens that some sites that `ngsub` has to contact are slow to answer, or are down altogether. This will not prevent you from submitting a job, but will slow down the submission. To speed it up, you may want to specify a shorter timeout (default is 40 seconds) with the `-t` option:

```
ngsub -t 5 myjob.xrsl
```

A default value for the timeout can be set by the first existing of:

1. `timeout` option in `client` section in `$HOME/.arc/client.conf`
2. `timeout` option in `common` section in `$HOME/.arc/client.conf`
3. `NGTIMEOUT` option in `$HOME/.ngrc` (ARC $\leq$ 0.5.48)
4. `timeout` option in `client` section in `$ARC_LOCATION/etc/arc.conf`
5. `timeout` option in `common` section in `$ARC_LOCATION/etc/arc.conf`
6. `timeout` option in `client` section in `/etc/arc.conf`
7. `timeout` option in `common` section in `/etc/arc.conf`

The user interface transforms input xRSL job description into a format that can be understood by the Grid Manager to which it is being submitted. By specifying the `-dumpxrsl` option, the transformed xRSL is written to stdout instead of being submitted to the remote site.

8.2.3 ngstat

The `ngstat` command is used for obtaining the status of jobs that have been submitted with ARC.

ngstat [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -clusters`		show information about sites (clusters)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-g, -giisurl`	*url*	URL of a central Information System server
`-G, -giislist`	*filename*	list of GIIS URLs
`-q, -queues`		show information about clusters and queues
`-l, -long`		long format (extended information)
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

The `ngstat` command returns the status of jobs submitted to the Grid.

> When several of the `-a`, `-i`, `-c`, `-C`, `-s` and `[job...]` command line options are specified, the comand returns information for **ALL** jobs that match either of the criteria (logical OR).
> For example, `ngstat -s FINISHED -c my.grid.org <jobid>` will return information about all finished jobs on the Grid, plus about all jobs (in any state) on the cluster `my.grid.org`, plus about the job `<jobid>`.

A job can be referred to either by the `jobID` that was returned by `ngsub` at submission time, or by its name if the job description specified a job name.

More than one `jobID` and/or job name can be given. If several jobs were submitted with the same job name, the status of all those jobs will be shown. If the `-i` option is used, the list of `jobID`s is read from a file with the specified name. By specifying the `-a` option, the status of all jobs of this user in the system will be shown.

> `ngstat` uses local list of jobs (see Section 8.2.11). If you are using several different machines to submit jobs, make sure to issue `ngsync` command (Section 8.2.1) after changing the machine, to synchronise the list of known jobs.
> `ngstat` can not return information about jobs that have been **erased**. The jobs are normally erased by the sites if they were not retrieved within 24 hours after job end.

A standard output produced by `ngstat` is rather short, informing the user only about the job status. Use `ngstat -l` to receive the complete job information as stored in the system.

The `-s` option can be used to select jobs in a specific state. For finished jobs, the `-s` option will match `FINISHED` or `FAILED` depending on whether the job finished successfully or not. These options can be repeated several times. For example, to check the status of jobs being on a particular stage of execution (e.g., only running jobs), use:

```
ngstat -s "INLRMS:R"
```

Diferent sites may report slightly diferent job states, depending on the installed software version. A summary of essential job states is:

ARC 0.3, ARC 0.4	ARC 0.5, ARC0.6	Description
	ACCEPTING	job has reached the site
ACCEPTED	ACCEPTED	job submitted but not yet processed
PREPARING	PREPARING	input files are being retreived
	PREPARED	input files are retreived
SUBMITTING	SUBMITTING	interaction with LRMS ongoing
INLRMS: Q	INLRMS:Q	job is queued by LRMS
INLRMS: R	INLRMS:R	job is running
	INLRMS:S	job is suspended
	INLRMS:E	job is finishing in LRMS
	INLRMS:O	job is in any other LRMS state
CANCELING	KILLING	job is being cancelled by user request
	EXECUTED	job is completed in LRMS
FINISHING	FINISHING	output files are being transferred
FINISHED	FINISHED	job is finished
	FAILED	job is finished with an error
	KILLED	job is cancelled by user request
DELETED	DELETED	job is removed due to expiration time

Detailed information on job states can be found in the Information System manual [4]. For ARC versions 0.3 and 0.4, most of the states may be prepended with the "`PENDING:`" message, if the job can not move from the state – for example, if the limit of jobs in the next state is reached. That is, `PENDING:PREPARING` means that the job **exited** the `PREPARING` state and is pending transfer to the next state, `SUBMITTING`. The `FINISHED` state may be followed by an error message if the job failed, starting with "`: FAILURE`" string. In ARC versions 0.5 and 0.6, new states are added instead, to better describe the situaton.

If the `-q` option is used, `ngstat` returns the status of the sites and queues available rather than of the jobs submitted. In this case the `-c` and `-C` options can be used to select or reject clusters for which the status should be returned.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

8.2.4 ngcat

It is often useful to monitor the job progress by checking what it prints on the standard output or error. The command `ngcat` assists here, extracting the corresponding information from the execution cluster and pasting it on the user's screen. It works both for running tasks and for the finished ones. This allows a user to check the output of the finished task without actually retreiving it.

ngcat [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of job IDs
`-c, -clusters`		show information about clusters
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-o, -stdout`		show the stdout of the job (default)
`-e, -stderr`		show the stderr of the job
`-f, -follow`		show tail of the requested file and follow its changes
`-l, -gmlog`		show the grid error log of the job
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

The `ngcat` command can return the standard output of a job (`-o` option), the standard error (`-e` option) and the errors reported by the Grid Manager (`-l` option).

> `ngcat` can only operate on jobs where either `stdout`, `stderr` and/or `gmlog` files were specified in the job description, respectively.

Option `-f` is handy for very long output files: it prints out only the last lines of the specified output, and follows its updates.

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 8.2.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

> When several of the `-a`, `-i`, `-c`, `-C`, `-s` and `[job...]` command line options are specified, the comand prints logs for **ALL** jobs that match either of the criteria (logical OR).
> For example, `ngcat -s FINISHED -c my.grid.org <jobid>` will print logs of all finished jobs on the Grid, plus of all jobs (in any state) on the cluster `my.grid.org`, plus of the job `<jobid>`.

8.2.5 ngget

To retrieve the results of a finished job, the `ngget` command should be used. It will download the files specified by the `outputfiles` attribute of job description to the user's computer.

ngget [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-dir`	*dirname*	download directory (the job directory will be created in this directory)
`-j, -usejobname`		use the jobname instead of the digital ID as the job directory name
`-keep`		keep files on gatekeeper (do not clean)
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

Only the results of jobs that have finished can be downloaded. The job can be referred to either by the `jobID` that was returned by `ngsub` at submission time, or by its name, if the job description contained a job name attribute.

The files to be retrieved by `ngget` should be described by the xRSL as follows:

```
(outputfiles=(file1 "")(file2 ""))
```

That is, while the filenames must be listed, no output destination should be requested. This will tell the Grid Manager not to erase the files after job's end and allow the ARC client to download them.

> Files specified as `stdout`, `sterr` and the `gmlog` directory are always treated as `outputfiles` by the system, which means you don't have to add them to the output files list.

By default, `ngget` will create in your current directory a new folder, by the same name as the remote session directory (typically, a numerical string). This new directory will contain all the files listed for download in your job description. If you would like to store the files in another location, use the `-dir` option. The option `-j` will assign your job name to the local directory with the output files (be careful not to call all the jobs same name when using this option).

> UI **does not clean up** such download areas, so please take care of doing this yourself.

A default value for the download directory other than the current working directory can be set by the first existing of:

1. `downloaddir` option in `client` section in `$HOME/.arc/client.conf`
2. `downloaddir` option in `common` section in `$HOME/.arc/client.conf`
3. `NGDOWNLOAD` option in `$HOME/.ngrc` (ARC $\leq$ 0.5.48)
4. `downloaddir` option in `client` section in `$ARC_LOCATION/etc/arc.conf`
5. `downloaddir` option in `common` section in `$ARC_LOCATION/etc/arc.conf`
6. `downloaddir` option in `client` section in `/etc/arc.conf`
7. `downloaddir` option in `common` section in `/etc/arc.conf`

> If your job produces many small output files, consider archiving them into a single file. Transfering a lot of small files at a time may temporarily consume all TCP ports allocated for data transfer on some servers, resulting in an error.

When the job result is retrieved, the session directory is **erased** from the execution machine, and the job ID is removed from your list of submitted jobs. If you however want to keep the job for a while, use the `-k` option.

> Beware that the job results will be **removed** by the Grid Manager at the execution machine 24 hours after job completion independently of whether you retrieved the results or not.

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 8.2.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

> When several of the `-a`, `-i`, `-c`, `-C`, `-s` and `[job...]` command line options are specified, the comand retrieves **ALL** jobs that match either of the criteria (logical OR).
> For example, `ngget -s FINISHED -c my.grid.org <jobid>` will retrieve all finished jobs on the Grid, plus all jobs (in any state) on the cluster `my.grid.org`, plus the job `<jobid>`.

8.2.6 ngkill

It happens that a user may wish to cancel a job. This is done by using the `ngkill` command. A job can be killed amost on any stage of processing through the Grid.

ngkill [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -clusters`		show information about clusters
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-keep`		keep files on gatekeeper (do not clean)
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0

`-x, -anonymous`	use anonymous bind for queries (default)
`-X, -gsi`	use GSI-GSSAPI bind for queries
`-v, -version`	print version information
`-h, -help`	print help page
Arguments:	
`job ...`	list of job IDs and/or jobnames

When option `-j` is used, and several jobs have the same name, **ALL such jobs will be cancelled!**.

> Job cancellation is an asynchronous process, such that it may take a few minutes before the job is actually cancelled.

When a job is successfully killed, it is **erased** from the remote site. Use `-keep` option to keep the output for eventual retrieval with `ngget` or `ngcat`.

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 8.2.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

> When several of the `-a`, `-i`, `-c`, `-C`, `-s` and `[job...]` command line options are specified, the comand kills **ALL** jobs that match either of the criteria (logical OR).
> For example, `ngkill -s INLRMS:R -c my.grid.org <jobid>` will kill all running jobs on the Grid, plus all jobs (in any state) on the cluster `my.grid.org`, plus the job `<jobid>`.

8.2.7 ngresub

Quite often it happens that a user would like to re-submit a job, but has difficulties recovering the original job description xRSL file. This happens when xRSL files are created by scripts on-fly, and matching of xRSL to the job ID is not straightforward. The utility called `ngresub` helps in such situations, allowing users to resubmit jobs known only by their job IDs.

> Only jobs where the `gmlog` attribute was specified in the job description can be resubmitted.

ngresub [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-k, -kluster`	`[-]`textemname	explicitly select or reject a specific site (cluster) as re-submission target
`-K, -Klustlist`	`[-]`textemfilename	list of clusters to select or reject as re-submission target
`-g, -giisurl`	*url*	URL of a central Information System server

`-G, -giislist`	*filename*	list of GIIS URLs
`-o, -joblist`	*filename*	file where the job IDs will be stored
`-dryrun`		add dryrun option to the xRSL
`-dumpxrsl`		do not submit – dump transformed xRSL to stdout
`-keep`		keep files on gatekeeper (do not clean)
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-U, -unknownattr`		allow unknown attributes in job description – pass them through
`-v, -version`		print version information
`-h, -help`		print help page
Arguments:		
`job ...`		list of job IDs and/or jobnames

If the original job description contained input file locations specified as relative paths (non-URLs), the command must be issued in the same directory as the original `ngsub` instruction.

It is important to distinguish `-c` and `-k` options (as well as `-C` and `-K`). The former stands for `-cluster` and is used to instruct `ngresub` to look for jobIDs and descriptions only at a given site (cluster), or exclude a cluster. This is convenient to use together with the `-a` option, in case you would like to re-submit all the jobs which were originally sent to a specific site (cluster). The latter option, `-k`, stands for `-kluster`, and should be used to specify preferred re-submission target. **Important** examples:

```
ngresub -c fire.ii.uib.no myjob1
```

will resubmit all jobs on `fire.ii.uib.no` and the job `myjob1` to any cluster (except the cluster at which the job is currently), while

```
ngresub -k fire.ii.uib.no myjob1
```

will resubmit the job `myjob1` from wherever it was to `fire.ii.uib.no` (note the usage of `-c` and `-k` options).

```
ngresub -c grid1.ua.org -k grid2.ub.org
```

will resubmit all your jobs from cluster `grid1.ua.org` to `grid2.ub.org`.

If the re-submission was successful, the original job will be removed from the remote cluster unless you specify the `-keep` option.
Old Grid Manager log (`gmlog`) records are not preserved and are not transferred to the new site..

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 8.2.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-o`, `-dryrun`, `-dumpxrsl`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

When several of the `-a`, `-i`, `-c`, `-C`, `-s` and `[job...]` command line options are specified, the comand resubmits **ALL** jobs that match either of the criteria (logical OR).
For example, `ngresub -s FAILED -c my.grid.org <jobid>` will resubmit all failed jobs on the Grid, plus all jobs (in any state) on the cluster `my.grid.org`, plus the job `<jobid>`.

8.2.8 ngclean

If a job fails, or you are not willing to retrieve the results for some reasons, a good practice for users is not to wait for the Grid Manager to clean up the job leftovers, but to use `ngclean` to release the disk space and to remove the job ID from the list of submitted jobs and from the Information System.

ngclean [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-f, -force`		removes the job ID from the local list even if the job is not found on the Grid
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

Only jobs that have finished can be cleaned.

Use `-f` option to remove jobs from your local list of jobs (`.ngjobs` file), if they can not be found in the information system and you are confident they are not needed anymore.

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 8.2.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

> When several of the `-a`, `-i`, `-c`, `-C`, `-s` and `[job...]` command line options are specified, the comand cleans **ALL** jobs that match either of the criteria (logical OR).
> For example, `ngclean -s FAILED -c my.grid.org <jobid>` will clean all failed jobs on the Grid, plus all jobs (in any state) on the cluster `my.grid.org`, plus the job `<jobid>`.

8.2.9 ngrenew

Quite often, the user proxy expires while the job is still running (or waiting in a queue). In case such job has to upload output files to a Grid location (Storage Element), it will fail. By using the `ngrenew` command, users can upload a new proxy to the job. This can be done while a job is still running, thus preventing it from failing.

If a job has failed in file upload due to expired proxy, `ngrenew` can be issued whithin 24 hours (or whatever is the expiration time set by the site) after the job end, which must be followed by `ngresume`. The Grid Manager will then attempt to finalize the job by uploading the output files to the desired location. In ARC versions prior to 0.6, `ngrenew` automatically resumed jobs failed in upload.

ngrenew [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

Prior to using `ngrenew`, be sure to actually create the new proxy, e.g.:

```
grid-proxy-init -valid 24:00
```

Here `-valid` specifies for how long the proxy will be valid. You can renew a VOMS-extended proxy as well (use `voms-proxy-init`).

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 8.2.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

When several of the `-a`, `-i`, `-c`, `-C`, `-s` and `[job...]` command line options are specified, the comand renews proxies for **ALL** jobs that match either of the criteria (logical OR).
For example, `ngrenew -s FAILED -c my.grid.org <jobid>` will renew proxies of all failed jobs on the Grid, plus of all jobs (in any state) on the cluster `my.grid.org`, plus of the job `<jobid>`.

8.2.10 ngresume

In some cases a user may want to restart a failed job, for example, when input files become available, or the storage element for the output files came back online, or when a proxy is renewed with `ngrenew`. This can be done using the `ngresume` command.

Make sure your proxy is still valid, or when uncertain, run `ngrenew` before `ngresume`. The job will be resumed from the state where it has failed.

ngresume [options] [job ...]

(ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 8.2.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 8.2.2.

> When several of the `-a`, `-i`, `-c`, `-C`, `-s` and `[job...]` command line options are specified, the comand resumes **ALL** jobs that match either of the criteria (logical OR).
> For example, `ngresume -s FAILED -c my.grid.org <jobid>` will resume all failed jobs on the Grid, plus all jobs (in any state) on the cluster `my.grid.org`, plus the job `<jobid>`.

8.2.11 Auxilliary files

ARC client keeps local job lists in two files: `$HOME/.ngjobs` and `$HOME/.arc/history`[¶].

`$HOME/.ngjobs` is a local list of the user's active jobs. When a job is successfully submitted, it is added to this list, and when it is removed from the remote site, it is removed from this list. This list is used as the list of all active jobs when the user specifies `-a` option to the various ARC user interface commands. For information about how to reconstruct this file in case it is damaged or you relocate to a different workstation, see section 8.2.1 about the `ngsync` command.

`$HOME/.arc/history` contains the `jobID`s of the jobs the user has submitted together with the time of submission. This file is purely informational.

[¶] In ARC $\leq$ 0.5.48, `$HOME/.nghistory`

Chapter 9

Data Management

At this stage, data management possibilities in an ARC-enabled system are limited to basic operations such as simple, one-by-one, file copy and removal, with eventual use of data indexing services to assist in arranging files in logical collections, keep track of their replicas, and simplify storage resource discovery.

9.1 Working with files

9.1.1 ngls

The `ngls` is a simple utility that allows to list contents and view some attributes of objects of a specified (by an URL) remote directory.

ngls [options] <URL>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:		
`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-l`		detailed listing
`-L`		detailed listing including URLs from which file can be downloaded
`-m`		list all available metadata attributes
`-r`	*level*	list recursively to given level
Arguments:		
`URL`		file or directory URL

This tool is very convenient not only because it allows to list files at a Storage Element or records in an indexing service, but also because it can give a quick overview of a job's working directory, which is explicitly given by job ID.

Usage examples can be as follows:

```
ngls rls://rc.host:38203/logical_file_name
ngls -l gsiftp://lscf.nbi.dk:2811/jobs/1323842831451666535
ngls -L srm://grid.uio.no:58000/srm/managerv1/johndoe/log2
```

Examples of URLs accepted by this tool can be found in Section 7.1.1, though `ngls` won't be able to list a directory at an HTTP server, as they normally do not return directory listings.

9.1.2 ngcp

The `ngcp` command is a powerful tool to copy files over the Grid. It is a part of the Grid Manager, but can be used by the User Interface as well.

ngcp [options] <source> <destination>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level: from -3 = quiet to 3 = verbose
`-y`	*cache_path*	path to local cache (use to put file into cache)
`-p`		use passive transfer (does not work if secure is on, default if secure is not requested)
`-n`		do not try to force passive transfer
`-i`		show progress indicator
`-u`		use secure transfer (insecure by default)
`-r`	*recursion_level*	operate recursively (if possible) up to specified level (0 - no recursion)
`-R`	*number*	how many times to retry transfer of every file before failing
`-t`	*time*	timeout in seconds (default 20)
`-f`		if the destination is an indexing service and not the same as the source and the destination is already registered, then the copy is normally not done. However, if this option is specified the source is assumed to be a replica of the destination created in an uncontrolled way and the copy is done like in case of replication. Using this option also skips validation of completed transfers.
`-T`		do not transfer file, just register it - destination must be non-existing meta-url

Arguments:

`source`	source URL
`destination`	destination URL

This command transfers contents of a file between 2 end-points. End-points are represented by URLs or meta-URLs. For supported endpoints please refer to Section 7.1.1.

`ngcp` can perform multi-stream transfers if `threads` URL option is specified and server supports it.

Source URL can end with `"/"`. In that case, the whole fileset (directory) will be copied. Also, if the destination ends with `"/"`, it is extended with part of source URL after last `"/"`, thus allowing users to skip the destination file or directory name if it is meant to be identical to the source.

> Since the job ID is in fact a `gsiftp://` URL of the job top directory, you can use `ngcp` to copy files from the job directory at any time.

Usage examples of `ngcp` are:

```
ngcp gsiftp://lscf.nbi.dk:2811/jobs/1323842831451666535/job.out \
          file:///home/myname/job2.out
ngcp gsiftp://aftpexp.bnl.gov;threads=10/rep/my.file \
          rc://;threads=4@grid.uio.no/lc=Collection,rc=Catalog/zebra4.f
ngcp http://www.nordugrid.org/data/somefile gsiftp://hathi.hep.lu.se/data/
```

9.1.3 ngrm

The `ngrm` command allows users to erase files at any location specified by a valid URL.

ngrm [options] <source>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:		
`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-c`		continue with meta-data even if it failed to delete real file
`-C`	*cache_data_path*	store cached data
Arguments:		
`source`		source URL

> A convenient use for `ngrm` is to erase the files in a data indexing catalog (RC, RLS or such), as it will not only remove the physical instance, but also will clean up the database record.

Here is an `ngrm` example:

```
ngrm rc://grid.uio.no/lc=Collection,rc=Catalog/badfile#\\
```

9.1.4 ngacl

This command retrieves or modifies access control information associated with a stored object if service supports GridSite GACL language [19] for access control.

ngacl [options] get|put <URL>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:		
`-d, -debug`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-v`		print version information
`-h`		short help
Arguments:		
`get`	*URL*	get Grid ACL for the object
`put`	*URL*	set Grid ACL for the object
`URL`		object URL; curently only gsiftp and sse URLs are supported

ACL document (an XML file) is printed to standard output when `get` is requested, and is acquired from standard input when `set` is specified*. Usage examples are:

```
ngacl get gsiftp://se1.ndgf.csc.fi/ndgf/tutorial/dirname/filename
ngacl set gsiftp://se1.ndgf.csc.fi/ndgf/tutorial/dirname/filename $<$ myacl
```

9.1.5 ngtransfer

The `ngtransfer` command initiates direct transfer of data between 2 servers (known as *third-party transfer*).

ngtransfer [options] <destination>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:		
`-s, -source`	*URL*	source file URL
`-d, -debug`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-v`		print version information
`-h`		short help
Arguments:		
`destination`		destination URL; currently only se:// and (gsi)ftp:// are supported

This command initiates file copy from multiple source instances to a destination URL. Destination of `(gsi)ftp` type accepts only similar kinds of sources. Destination can also be URL of an Indexing Service. In such a case, real destinations with suitable protocols are chosen, when available. Requests are sent to the corresponding services/servers to initiate file transfer from one of the sources.

> Absence of `-s` option is treated as a file replication request. In this case, destination must be an Indexing service or an SRM.

Following source URL types are supported: `http`, `https`, `httpg`, `ftp`, `gsiftp`, `rc`, `rls`, `se`, `srm`, `fireman` (see Section 7.1.1 for URL details).

Example:

```
ngtransfer -s http://www.host1.org/dat1  -s gsiftp://host2.org/dir/dat1 \
           se://se.host.org/se_service?new_file_lfn
```

9.1.6 ngstage

The `ngstage` command is used to request file staging from tapes to front-end disks. Since jobs can not read data from tapes directly, if an input file is on a tape, the job most ikely will fail for a timeout, because tape staging process is a rather lengthy one (can be few hours). To avoid these timeout failures, `ngstage` should be used before submitting jobs that are known to use data archived on tapes.

ngstage [options] <URL(s)>

(ARC 0.6, ARC 0.8)

Options:

*In ARC $\leq$ 0.5.28, `set` shoud be used instead of `put`

`-q`	*requestID*	query the status of a request. Use `-d 1` in addition to show a per-file status.
`-c`	*requestID*	cancel a request
`-D`		do a dry run, listing the files that would be staged
`-d`	*debuglevel*	debug level from 0 to 3 (default 0). Setting to 1 when listing a request status will show the status of each file in the request.
`-r`	*level*	if copying fileset/directory recurse to the specified level. If any given URL is a directory a level of at least 1 must be specified.
`-s`	*URL*	the endpoint of the staging service. This must be given with `-q`, `-c` and `-l` options.
`-t`	seconds	timeout in seconds. This is not a timeout for staging the file(s), but a timeout for calls to the service performing the staging. A timeout of this kind does not necessarily mean that the staging request has failed.
`-v`		print version information
`-h`		short help
Arguments:		
`URL`		file URL; currently only srm:// meta-protocol is supported

The command will return a request ID that should be used to monitor the status (with `-q` option) or cancel the request (`-c` option). A list of currently active requests can be found using the `-l` option. If either of these three options are used, URL of the file server performing the staging must also be specified, through the `-s` option, this usually consists of the protocol (`srm://`) and hostname of the URL(s) passed in the inital request.

> Every URL in the request must have the same host name, as the command sends request to one file server at a time.

Wildcards are allowed at the end of a URL, and URLs can be directories, but `-r` must be used if the contents of a directory are to be staged. If a given URL is a directory, `-r 1` will stage all files in that directory.

Example:

```
ngstage srm://srm.ndgf.org/pnfs/ndgf.org/data/atlas/tape/file.1\\
ngstage -q -2139909111 -s srm://srm.ndgf.org
```

This requests a file to be staged and queries request status.

> After the first time a successful request is queried, it is internally marked as aborted by the system. For any subsequent queries it is not possible to obtain the per-file status.

9.2 Replica Location Service in Examples

This section lists basic examples showing how to access and manipulate the records in the Globus Replica Location Server (RLS) [18]. Only authorised RLS managers can perform operations that require write access.

ARC is interfaced to the Globus RLS[†]. RLS relies on a simple relational database that does not support data sets or collections. An RLS record consists of a logical file names (LFN) which corresponds to one or

[†]Globus RLS is substantially different from EDG/LCG RLS; the latter is not described here and is not interfaced to ARC

more physical files (PFN). Each logical file has also an associated set of attributes, such as size, checksum, GUID etc.

LFNs are usually just file names, e.g.

```
dc2.003007.simul.A1_z_ee._00001.pool.root
```

because the Grid can not guess what logical name one would assign to a file.

PFNs are transport, "normal" URLs of the files, e.g.

```
gsiftp://dc4.uio.no/dc2/dc2.003007.simul.A1_z_ee._00002.pool.root.4
```

Each LFN can have several PFNs (*replicas*).

> NorduGrid RLS has additional special storage service records that represent *storage facilities*.

Such special records allow to index storage facilities and help the Grid to locate destinations for output files. These storage service records have a special format: they have fixed "LFN" `__storage_service__`. Storage service PFNs include complete path to the expected file destination, thus resembling a collection, e.g.

```
gsiftp://hive-storage.unicc.chalmers.se:2811/storage/data
```

Regular users are not advised to create or remove these records. Certain ARC services (e.g., the Smart Storage Element) insert these records automatically as a part of the registration process.

There are two ways of working with RLS: interactively, which means you only authenticate yourself once per session, or issuing a separate command for every operation.

START INTERACTIVE SESSION:

```
globus-rls-cli -u rls://rls.nordugrid.org:39281
```

Examples below show both interactive (marked **I:**) and command-line (**C:**) variants of every instruction; server `rls://rls.nordugrid.org:39281` is used as an example, replace it with your favorite server name:port combination.

Below, strings `lfn` or `pfn` have to be typed just like this, "`lfn`" or "`pfn`", while `<lfn>` and `<pfn>` have to be replaced with actual LFNs and PFNs.

9.2.1 Show all possible attribute names for LFNs

```
I:  attribute show - lfn
C:  globus-rls-cli attribute show - lfn rls://rls.nordugrid.org:39281
```

9.2.2 Show the value of an attribute for an LFN

Suppose LFNs have an associated attribute "`dq_guid`". To find its value for a given LFN, do:

```
I:  attribute query <lfn> dq_guid lfn
C:  globus-rls-cli attribute query <lfn> <attribute-name> lfn \
          rls://rls.nordugrid.org:39281
```

9.2.3 Show the values of all attributes for a LFN

```
I:  attribute query <lfn> - lfn
C:  globus-rls-cli attribute query <lfn> - lfn \
          rls://rls.nordugrid.org:39281
```

9.2.4 Show all the storage services

```
I:  query lrc lfn __storage_service__
C:  globus-rls-cli query lrc lfn __storage_service__ \
          rls://rls.nordugrid.org:39281
```

9.2.5 Show PFNs for LFN

```
I:  query lrc lfn <lfn>
C:  globus-rls-cli query lrc lfn <lfn> rls://rls.nordugrid.org:39281
```

9.2.6 Show LFN for a PFN

```
I:  query lrc pfn <pfn>
C:  globus-rls-cliquerylrcpfn<pfn>rls://rls.nordugrid.org:39281
```

9.2.7 Wildcard version of the above

```
I:  query wildcard lrc lfn *3014*
C:  globus-rls-cli query wildcard lrc lfn '<pattern>' \
          rls://rls.nordugrid.org:39281
```

9.2.8 Create an LFN-PFN pair

This example shows how to create a new record (LFN-PFN pair); it implies that LFN did not exist before:

```
I:  create <lfn> <pfn>
C:  globus-rls-cli create <lfn> <pfn> rls://rls.nordugrid.org:39281
```

9.2.9 Add a PFN to an already existing LFN

This is an action needed when a new replica of a file is created.

```
I:  add <lfn> <pfn>
C:  globus-rls-cli add <lfn> <pfn> rls://rls.nordugrid.org:39281
```

9.2.10 Delete a PFN from an LFN

This action is needed when a replica is removed from the Grid.

> Note: if the last PFN is removed, the LFN is also removed, so there's no reverse of the create operation!

```
I:  delete <lfn> <pfn>
C:  globus-rls-cli  delete <lfn> <pfn> rls://rls.nordugrid.org:39281
```

9.2.11 Define an LFN attribute in the RLS server

This action adds new attribute to all the records (LFNs). Initial value of the attribute is empty (NULL).

> Note: `<attribute-type>` usually is "`string`".

```
I:  attribute define <attribute-name> lfn <attribute-type>
C:  globus-rls-cliattributedefine<attribute-name>lfn\
          <attribute-type>rls://rls.nordugrid.org:39281
```

9.2.12 Add an attribute with a value to an LFN

This action adds an attribute to an LFN and sets it to a given value.

```
I:  attribute add <lfn> <attribute-name> lfn <attribute-type> <attribute-value>
C:  globus-rls-cliattributeadd<lfn><attribute-name>lfn\
          <attribute-type><attribute-value>rls://rls.nordugrid.org:39281
```

9.2.13 Add a Storage Element

```
I:  add __storage_service__ gsiftp://se1.abc.def.dk:2811/storage/data
C:  globus-rls-cliadd__storage_service__\
          gsiftp://se1.abc.def.dk:2811/storage/datarls://rls.nordugrid.org:39281
```

9.2.14 Remove a Storage Element

```
I:  delete __storage_service__ gsiftp://se1.abc.def.dk:2811/storage/data
C:  globus-rls-clidelete__storage_service__\
          gsiftp://se1.abc.def.dk:2811/storage/datarls://rls.nordugrid.org:39281
```

9.3 Replica Catalog in Examples

The NorduGrid Replica Catalog is nothing but slightly patched Globus Replica Catalog [17]. The support for it has been discontinued by Globus a while ago, and NorduGrid uses it just as a temporary solution for the distributed data catalogue. This Section only presents some usage examples and does not go into details of the Replica Catalog implementation, installation and support.

> Only authorised RC managers can perform operations which require write access. For details, contact `nordugrid-support@nordugrid.org`.

The Globus RC is an hierarchical database: the **catalogue** consists of **collections**, each collection consists of **logical file names** which correspond to one or more **physical files** at pre-defined **locations**. To register an arbitrary file in a RC, the following steps should be performed:

1. Ensure the collection exists
2. Ensure the physical location of the file is registered as one of the locations for the collection
3. Enter the file record into the RC, specifying the location and the file name **RELATIVE TO THE LOCATION**

The limitation of the Globus RC is that it is impossible to assign arbitrary logical file names: a logical file name **must** be the actual file name, eventually prepend with the path relative to the registered location.

9.3.1 Define who you are

This is the strongly recommended first step, simplifying authentication at the NorduGrid Replica Catalog

```
export GLOBUS_REPLICA_CATALOG_MANAGER="/O=Grid/O=NorduGrid/OU=nordugrid.org/CN=Jane Doe"
```

Or, for C shells:

```
setenv GLOBUS_REPLICA_CATALOG_MANAGER "/O=Grid/O=NorduGrid/OU=nordugrid.org/CN=Jane Doe"
```

The quoted string `/O=Grid/O=NorduGrid/OU=nordugrid.org/CN=Jane Doe` is your identity, as specified in the Subject of your Grid certificate. Use `grid-cert-info` command (Section 4.2.7) to find it.

9.3.2 Create a collection

```
globus-replica-management \
    -collection ldap://grid.uio.no/lc=A_Collection,rc=NorduGrid,dc=nordugrid,dc=org \
    -create
```

Here, `grid.uio.no` is the server address for the `NorduGrid` catalogue, and `A_Collection` is your new collection name. Since the RC is an LDAP database, collection URL must contain the LDAP-complying Distinguished Name, which includes the full NorduGrid RC name space (`dc=nordugrid,dc=org`). Default port number is 389, but if it is different, it should be specified in a standard IP manner: after the server address, separated by a column.

9.3.3 Add a location to a collection

```
globus-replica-management \
    -collection ldap://grid.uio.no/lc=A_Collection,rc=NorduGrid,dc=nordugrid,dc=org \
    -location CASTOR \
    -create gsiftp://wacdr002d.cern.ch/castor/cern.ch/atlas/transfer/dc1/
```

The location name `CASTOR` should later be used as a mount point, instead of the physical URL `gsiftp://...`

9.3.4 Upload and register a file to a collection

You must have a valid Grid proxy (see Section 6.1) to perform this operation

```
ngcopy \
  gsiftp://grid.quark.lu.se/ATLAS/mydata.zebra \
  rc://grid.uio.no/lc=A_Collection,rc=NorduGrid,dc=nordugrid,dc=org/yourdata.zebra
```

In the example above, the first available registered location will be used to store the file. To specify an explicit destination, do:

```
ngcopy \
  gsiftp://grid.quark.lu.se/ATLAS/mydata.zebra \
  rc://CASTOR@grid.uio.no/lc=A_Collection,rc=NorduGrid,dc=nordugrid,dc=org/yourdata.zebra
```

9.3.5 Register existing at a location file to a collection

File registration includes three steps:

1. Create a logical file entry in the collection:

```
globus-replica-catalog \
    -host ldap://grid.uio.no/lc=A_collection,rc=NorduGrid,dc=nordugrid,dc=org \
    -logicalfile path/file1.zebra \
    -create size 0
```

2. Verify that file location (mount point) is registered in the collection (see Sections 9.3.8 and 9.3.9), otherwise it has to be added (see Section 9.3.3).

3. Register the physical location(s):

```
globus-replica-management \
    -collection ldap://grid.uio.no/lc=A_collection,rc=NorduGrid,dc=nordugrid,dc=org \
    -location CASTOR \
    -files filelist \
    -register
```

If the file size is unknown, specify "0" as shown above. Logical entries can only be added one by one, while physical location(s) must be added via a `filelist`, one line per file:

```
path/file1.zebra
path/file2.zebra
```

> Note the prepended `path`, relative to the registered location: it should be possible to construct a correct URL as `${LOCATION}${FILENAME}`.

9.3.6 Remove and unregister a file

```
ngremove \
   rc://grid.uio.no/lc=A_Collection,rc=NorduGrid,dc=nordugrid,dc=org/yourdata.zebra
```

9.3.7 Remove a location from a collection

```
globus-replica-management \
    -collection ldap://grid.uio.no/lc=A_collection,rc=NorduGrid,dc=nordugrid,dc=org \
    -location CASTOR \
    -delete
```

9.3.8 Find locations (URL prefixes) for a collection

```
ldapsearch -h grid.uio.no -p 389 \
   -b "lc=A_Collection,rc=NorduGrid,dc=nordugrid,dc=org" \
   "(&(objectclass=GlobusReplicaLocation)(uc=*))" uc
```

or, using the Globus interface to LDAP:

```
globus-replica-catalog \
    -host ldap://grid.uio.no:389/lc=A_Collection,rc=NorduGrid,dc=nordugrid,dc=org \
    -collection \
    -list-locations uc
```

9.3.9 Find a URL prefix for a location known by name

```
globus-replica-catalog \
    -host ldap://grid.uio.no:389/lc=A_Collection,rc=NorduGrid,dc=nordugrid,dc=org \
    -location CASTOR \
    -list-attributes uc
```

Chapter 10

The Grid Monitor

The Grid Monitor is perhaps the most convenient utility that allows users and sysadmins alike to check and monitor the status of the ARC Grid facility, status of each job, display user-specific information and otherwise obtain in an intuitive way the information stored in the Information System [4]. It is implemented as a set of pop-up Web pages, connected by hyperlinks, making it easy to browse and navigate. Some limited searching capacity is also available, through the "Match-it-yourself" module.

The structure of the Grid Monitor to great extent follows that of the Information System [4]. The basic objects are defined by the following schema's objectclasses:

- nordugrid-cluster: a cluster
- nordugrid-queue: a queue at the cluster, accessible by the ARC users
- nordugrid-job: a Grid job, associated with a queue
- nordugrid-authuser: a user, authorized to submit jobs to a given queue

The Grid Monitor also uses the NorduGrid Virtual Organisation (VO) organisationalPerson and Storage Element nordugrid-se objectclasses, and their attributes.

For each objectclass, either an essential subset of attributes, or the whole list of them, is presented in an easily accessible inter-linked manner. This is realized as a set of windows, each being associated with a corresponding module. There are nine major modules :

1) An overall Grid Monitor
2) Cluster Description
3) Queue Details
4) Job Information
5) User Information
6) Attributes Overview
7) Customizeable Display Tool ("Match-it-yourself")
8) List of Storage Facilities
9) List of Users

Each module displays both dynamic and static information: for example, a queue name is static, while the amount of running jobs in this queue is dynamic. Most of the displayed objects are linked to appropriate modules, such that with a simple mouse click, a user can launch another module, expanding the information about the corresponding object or attribute. Each such module opens in an own window, and gives access to other modules in turn, providing thus rather intuitive browsing.

In what follows, these modules are described in detail, an overview of their functionality is given and usage hints are provided.

10.1 The Grid Monitor

The basic module, providing access to the most required information, is the Grid Monitor, showing the overall status of the system. It serves as a starting point for browsing the system information. The purpose of this module is to give a quick overview of the current status of the ARC Grid by showing the list of the available clusters and the most essential information about them: an alias, number of working processors, number of occupied processors and number of queueing jobs. In the current implementation, the main Grid Monitor window contains also the link to the user base of the ARC Grid. Figure 10.1 shows a screenshot of the running monitor. All the information shown is dynamic, including organizational names (countries in this case).

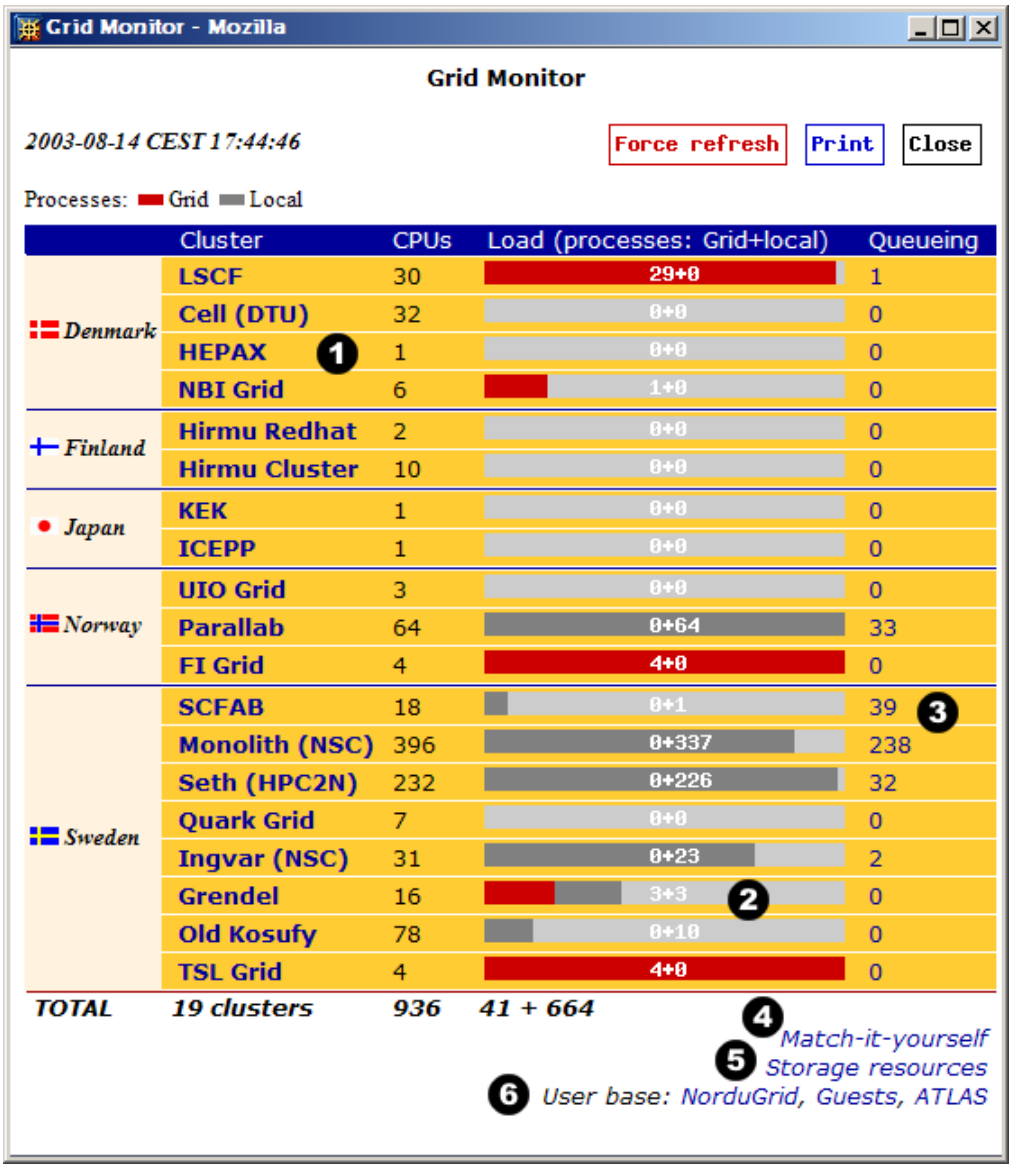

Figure 10.1: The Grid Monitor

In Figure 10.1, the numbered tags indicate clickable objects as explained below:

1) Cluster: a cluster alias, linked to the cluster description module (Section 10.2), which provides complete information about the current status of a cluster.
2) Load: a graphical and numeric representation of the cluster load, showing both Grid- and non-Grid (submitted locally) running processes. Red bar shows percentage of Grid processes, while the grey bar shows total relative occupancy of a cluster. Numbers indicate the absolute amount of running processes, with first figure corresponding to the Grid, and second - to the non-Grid ones. It should

be noted that number of processes does not necessarily correspond to the number of running jobs: a parallel job can occupy several processors. By clicking on a bar, a user accesses the list of all Grid jobs running on a cluster (Section 10.4).

3) Queueing: number of queueing jobs, which includes jobs queued in an LRMS as well as those being pre-processed by the Grid Manager [3]. Only jobs which can be potentially executed in a Grid queue are counted. The number is linked to the same module as the Load item, with the only difference that it displays the list of the Grid-queued jobs. Note that although non-Grid jobs are counted in the total number of queued jobs, they can not be listed by the Grid Monitor, as they do not provide any information to the Grid Information System.
4) Match-it-yourself: link to the "Match-it-yourself" interface (Section 10.7) which allows users to compose non-standard monitor requests.
5) Storage resources: link to the list of available storage resources (Section 10.8).
6) User base: several auxiliary links, providing an access to the VO-listing module (Section 10.9). The main purpose of this link is to provide an easy access to the user-specific information, such as the list of submitted jobs and available resources.

10.2 Cluster Description

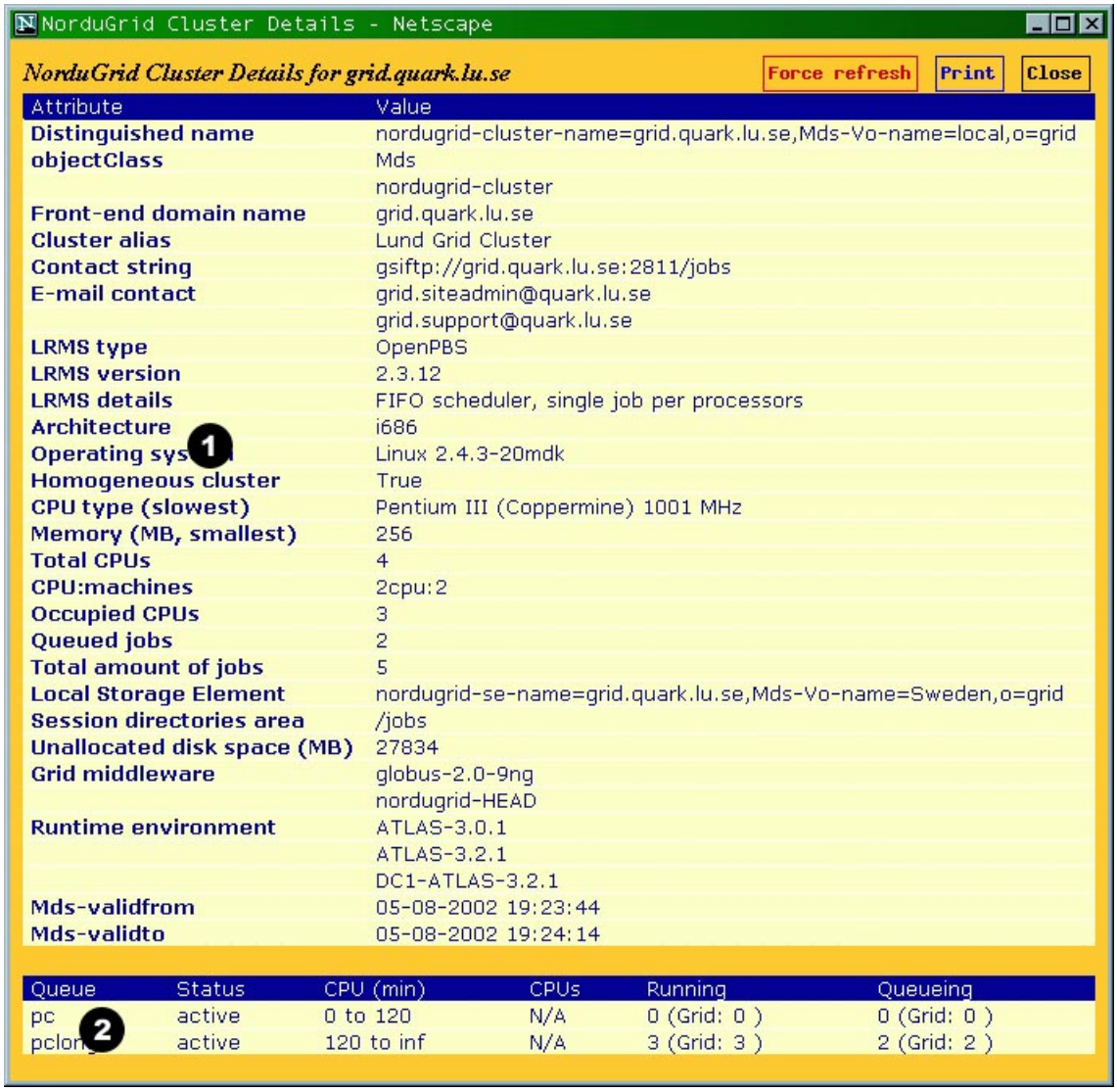

Figure 10.2: ARC-enabled cluster details

The cluster description module displays all the cluster attributes stored in the MDS, as well as most relevant information about the queues, accessible by the Grid users. The window thus contains two lists, as shown in Figure 10.2:

1) Attributes: this is a dump of all the attributes of the nordugrid-cluster objectclass, dynamic and static ones. Such attributes as cluster alias, or domain name, are static; others are dynamic, with the values obtained by the MDS from the information providers: e.g., total CPU number, amount of jobs, or available disk space. More details about these attributes can be found in the Grid Information System description [4]. Each attribute (apart of the MDS time stamps) is linked to the Attributes Overview module (Section 10.6), such that clicking on an attribute name brings the list of the values of this particular attribute on all the ARC-enabled clusters. For instance, this is the most convenient way to browse available disk space or runtime environment values over the system.
2) Queues: the list of queues at a given cluster, accessible by the Grid users. While the detailed list of queue attributes and corresponding jobs can be obtained by clicking on a queue name (see Queue Details module description, Section 10.3), the most essential parameters are listed already in the Cluster Description module. They are: queue name, queue status, queue length (minimal and maximal), number of CPUs assigned to a queue (if available), and number of running and queued jobs. Since queues can be shared between Grid and local users, the total number of jobs is shown, with the number of Grid jobs in parentheses.

The Cluster Description module is linked from most other modules (except List of Users): clicking on the domain name of a cluster opens the Cluster Description window.

10.3 Queue Details

In the Grid Information System, the nordugrid-queue objectclass is described by a set of queue-specific attributes, and has two sub-trees: nordugrid-job and nordugrid-authuser. This structure reflects the fact that users are not implicitly authorized to submit jobs to any queue. However, the list of users allowed to a specific queue is a fairly static information, and thus is beyond the scope of the Grid Monitor*.

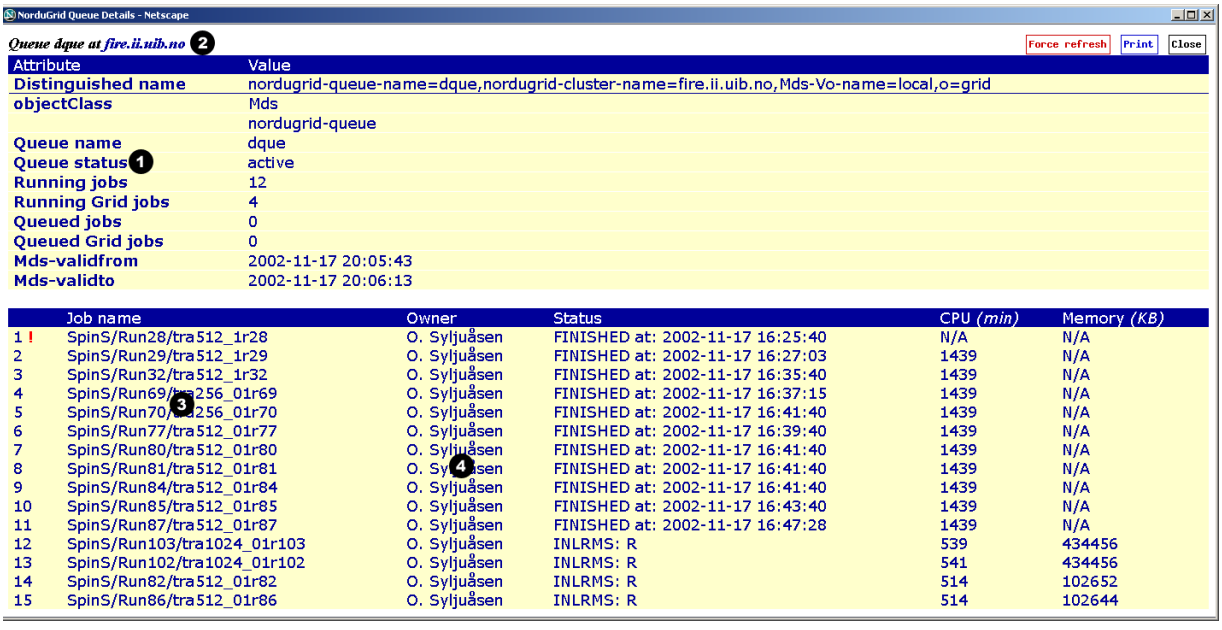

Figure 10.3: Grid queue details

The Queue Details module provides the list of the queue attributes and of all the jobs scheduled (running or waiting) to this queue. Figure 10.3 shows the Grid queue description window, with clickable fields marked by numbered tags as follows:

*List of queues available for a given user can be obtained through the User Information module.

1) Attributes: the dump of the queue attributes. Just like the cluster attributes (Section 10.2), they can be both static and dynamic. Every attribute is linked to the Attributes Overview module (Section 10.6), which allows the values of each attribute across the entire ARC system to be browsed.
2) Cluster name: each queue is associated with the cluster whose name is shown at the top of the window. Clicking the cluster name brings up the Cluster Description window (Section 10.2).
3) Job name: from the Queue Details window, users can get access to detailed information about every job in the queue by clicking the job name. Each job name is linked to the Job Information module, described in Section 10.4.
4) Owner: The Grid authentication mechanism allows every job to be associated to a corresponding user, even though an actual Unix account owner may be a generic "griduser". The Grid Monitor uses this feature to display explicitly each job owner. In the Queue Details window (as in all other modules), the user's name is linked to the User Information module (Section 10.5), which displays all the resources available for a given user, as well as the list of the user's jobs.

The Queue Information module is accessible via links to queue names in the Cluster Information (Section 10.2), Job Information (Section 10.4), User Information (Section 10.5) and Attributes Overview (Sec 10.6) modules.

10.4 Job Information

The Job Information module is activated on three different occasions:

- To display a list of all running Grid jobs on a cluster
- To display a list of all queued Grid jobs on a cluster
- To show the full information on a given job

Lists of running and queued jobs are accessible from the top Grid Monitor window (Section 10.1) by clicking the corresponding fields (marked 2 and 3 in Figure 10.1). As shown in Figure 10.4, such a list contains not only job names, but also their respective owners, status (as returned by the Grid Manager), execution time (in case of running jobs), and the submission queue.

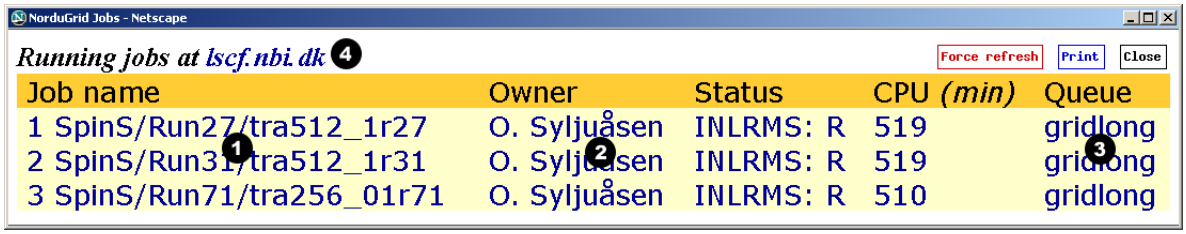

Figure 10.4: Grid job list

Most of the fields in a job list window are linked to the corresponding monitor modules, giving access to more detailed information:

1) Job name: just like in the Queue Details window (Section 10.3), the job name is linked to the Job Information window, described below. However, while the Queue Details module lists the jobs in a given queue, the Job Information window gives an overview of all the Grid jobs on a cluster.
2) Owner: this field is also identical to the one in the Queue Details window: the user's name is linked to the User Information module (Section 10.5), which displays all the resources available for a given user and the list of the user's jobs.
3) Queue: the name of the queue is liked to the Queue Details window (Section 10.3), which gives a snapshot of the queue status, including all the Grid jobs submitted to a particular queue – running or waiting.
4) Cluster name: clicking on the cluster name brings up the Cluster Description window (Section 10.2), which gives a general overview of a given cluster and the status of its queues (those available for the Grid users).

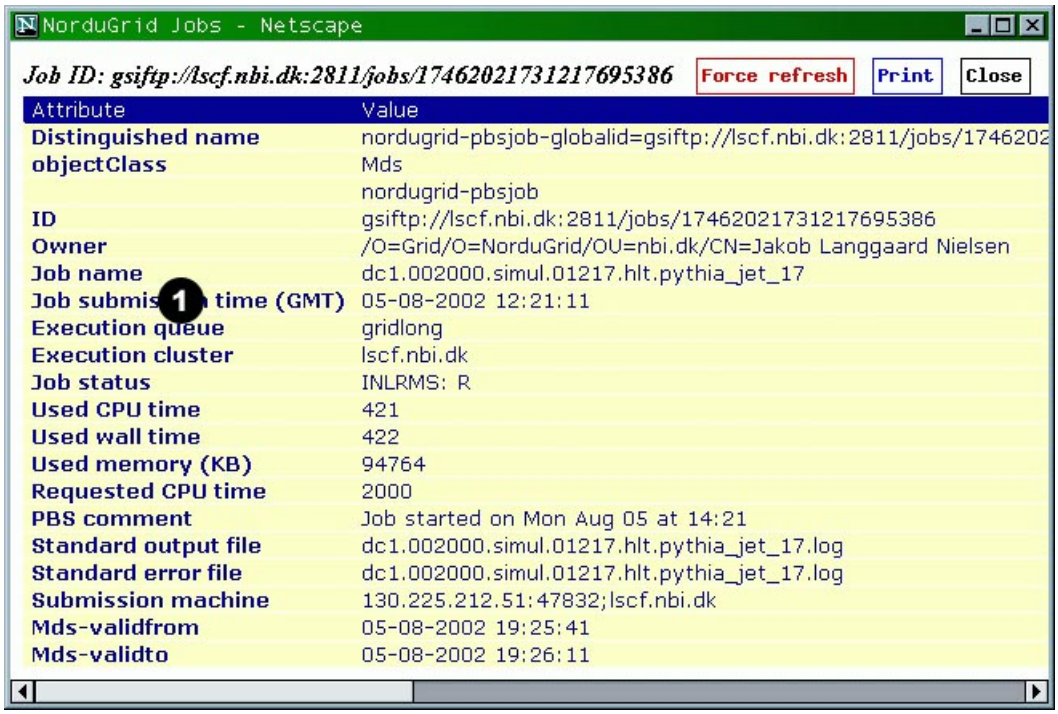

Figure 10.5: Grid job statistics

The job information window is invoked by clicking on a job name in any Grid Monitor window which lists jobs. It is handled by the same module which produces running/queued job list, and contains simple dump of all the available job attributes (see Figure 10.5). Just like in the Cluster Description and Queue Description windows, each attribute is clickable (as indicated by a tag numbered 1 in Figure 10.5), and is linked to the Attributes Overview module (Section 10.6). This is a convenient way to compare jobs that reside on the system.

10.5 User Information

The User Information module of the Grid Monitor gives access to all the available information related to a given user. This includes the list of available resources (queues, processors and disk space), and the list of user jobs residing on the system at the time of query. To collect this information, the whole system has to be queried, therefore invocation of this module typically takes quite a bit of time (at least compared to most other modules).

Figure 10.6 shows a typical User Information window, where the numbered fields are linked to other Grid Monitor modules:

1) Job name: this field is linked to the Job Information window (Section 10.4), providing access to the detailed information on a given job. Unlike the Job Information or Queue Information modules, which list jobs local to a cluster, the User Information module collects all the jobs submitted by a given user to the whole system.
2) Cluster: since the User Information window displays all the jobs associated with a given user, the description of each respective cluster is available by clicking the cluster name. This brings up a cluster description window, described in Section 10.2.
3) Queue: this field is linked to the Queue Details module (Section 10.3), thus giving access to the information about the status of the relevant queue.
4) Cluster: the upper part of the User Information window lists the Grid resources, available for a user. Each cluster, to which a user is authorized to submit jobs, is indicated by its name. Cluster names are linked to the Cluster Description window (Section 10.2), giving detailed information on available resources.

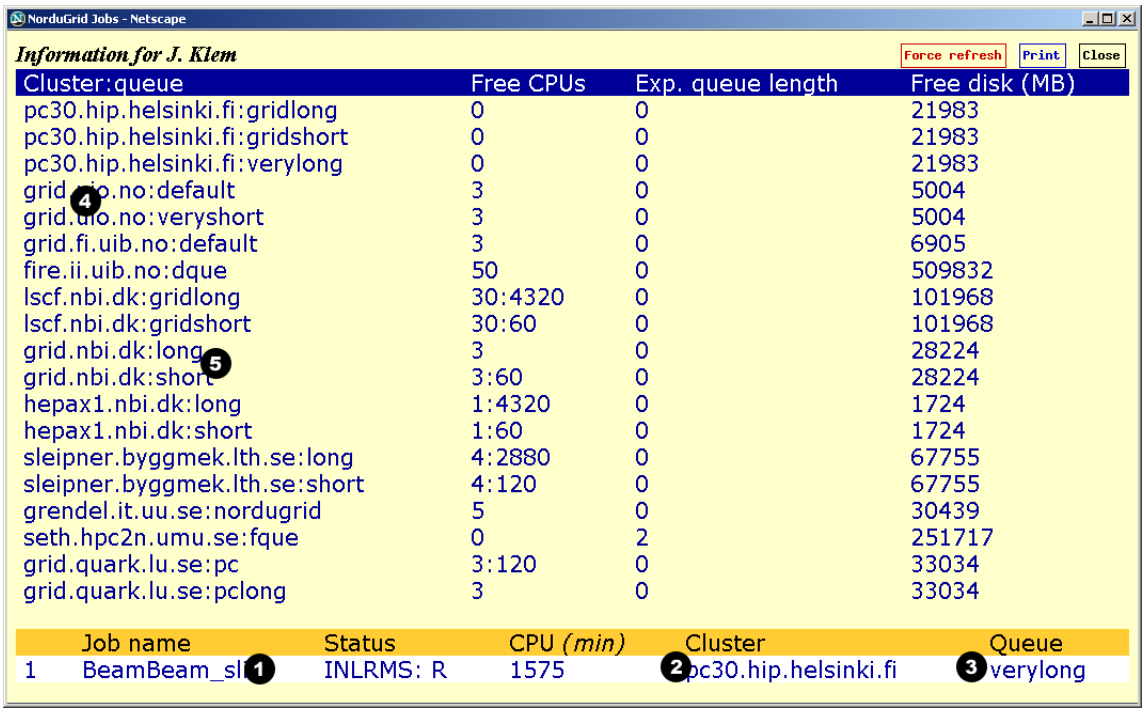

Figure 10.6: Grid user information

5) Queue: since users authorization may be not only cluster-based, but also queue-based, the allowed queue information can be accessed by clicking a queue name. This brings up the Queue Details window, described in Section 10.3.

The simplest way to access the User Information window is via the List of Users (Section 10.9), although it can be invoked from any Grid Monitor window where a user name is displayed (e.g., a Job Information or a Queue Details window).

10.6 Attributes Overview

As it was mentioned above, every ARC objectclass attribute, appearing in a Grid Monitor window, is linked to the Attributes Overview module, which queries all the relevant objects on the system and delivers a comparative list of the attributes. Similarly to the User Information module, querying all the Grid resources takes somewhat long time, as the Grid Monitor does not have its own cache.

This module can also be accessed via the "Match-it-yourself" interface (Section 10.7). In this case, it can list as many attributes as specified by a user request, eventualy applying the user selection criteria.

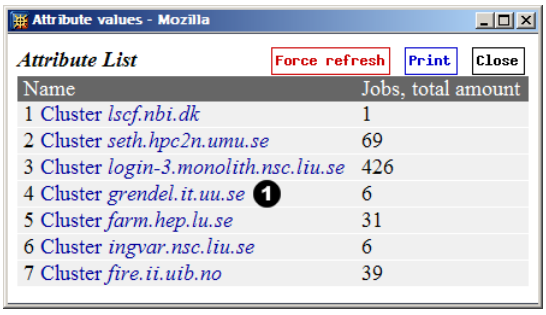

Figure 10.7: ARC objects grouped by attribute

Figure 10.7 shows a typical result of the Attributes Overview query: in this example, the nordugrid-cluster attribute "Jobs, total amount" was queried, and a comparative list of results returned. The Resource field

(indicated by the tag 1) depends on the nature of the attribute, and can be either of:

- cluster name, linked to the Cluster Description module,
- cluster name and queue name, linked to the Cluster Description and Queue Details modules respectively,
- job ID string (see ref.[3] for details), linked to the Job Information module.

10.7 "Match-it-yourself"

The "Match-it-yourself" display tool is a customizeable interface to the Attributes Overview module (Section 10.6). It allows users to chose which attributes of an object to display, optionally applying filters. While the other Monitor windows display a pre-defined set of data, this module gives an advanced user the possibility to build a customized request to the Information System.

An example use case for this interface could be a user desiring to view a list of his running (but not queued or finished) jobs, complete with used CPU and wall time, memory and job name. The "Match-it-yourself" tool would be then invoked for the job object, and the display request would contain Name, Used CPU time, Used wall time, Used memory (KB), and Status – the latter with a filter Status = INLRMS: R.

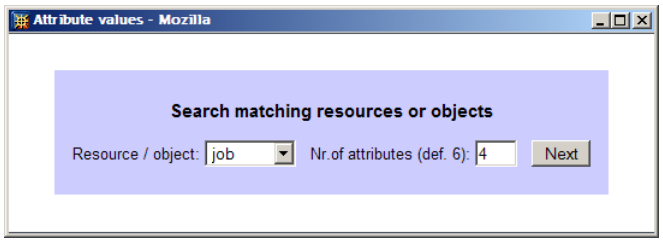

Figure 10.8: Object class selection window

Figure 10.8 shows the first screen of the "Match-it-yourself" interface, which welcomes users to select the object class to work with, and the amount of attributes to be displayed. When not sure about the latter, users should specify a top estimate – unused fields will be ignored in further searches.

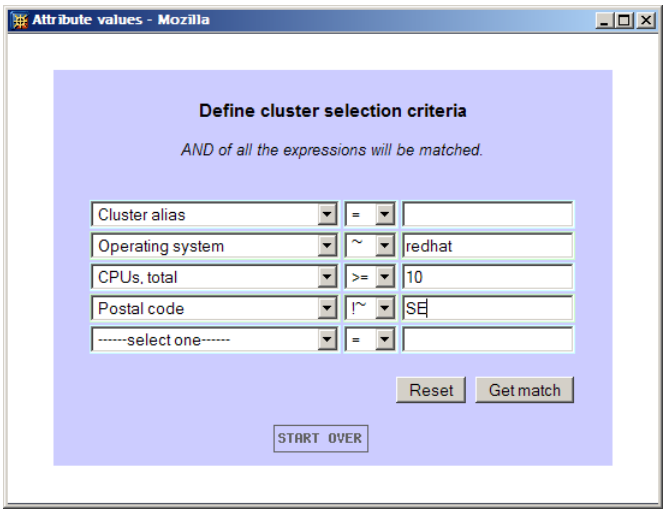

Figure 10.9: Attribute selection window

Figure 10.9 is a snapshot of the screen where the atttributes to display and their selection criteria are specified. If a user wishes to display an attribute for all the objects, independently of its value, the rightmost field may be either kept empty, or filled with an asterisk (*), while the middle field should be set to "=". Whenever a filter has to be applied, an operator should be selected in the middle column, and a match string

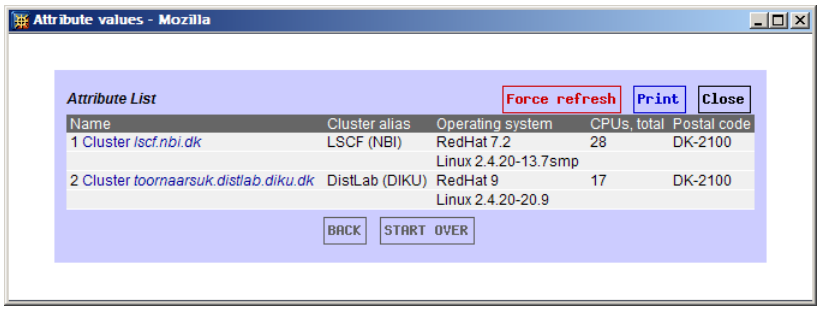

Figure 10.10: Customized cluster information display

specified in the rightmost field. For example, if only clusters containing "NBI" in their domain names have to be shown, the attribute filter would be Front-end domain name ~ nbi. Matches are case-insensitive.

Figure 10.10 is the result of the search according to the criteria defined in the exampel in Figure 10.9. Three filters were applied: on operating system attribute, total number of CPUs and postal code (in this case we were selecting any cluster which is not in Sweden). Since we wanted to display each cluster's alias as well, this attribute was added to the selection, but with a "match everything" scope. The attribute matching method is exactly the same as used by the Attributes Overview module (Section 10.6), and it re-uses the screen layout shown in Figure 10.7.

10.8 Storage Resources

Although there is no well-defined Storage Element concept in the NorduGrid, some information about the storage resources can be found in the Information System. The Storage Resources module, linked from the main Monitor window, displays all the available information for those Storage Elements which publish it. Particularly important is the base URL, which specifies the Grid mount point that could be used in job descriptions.

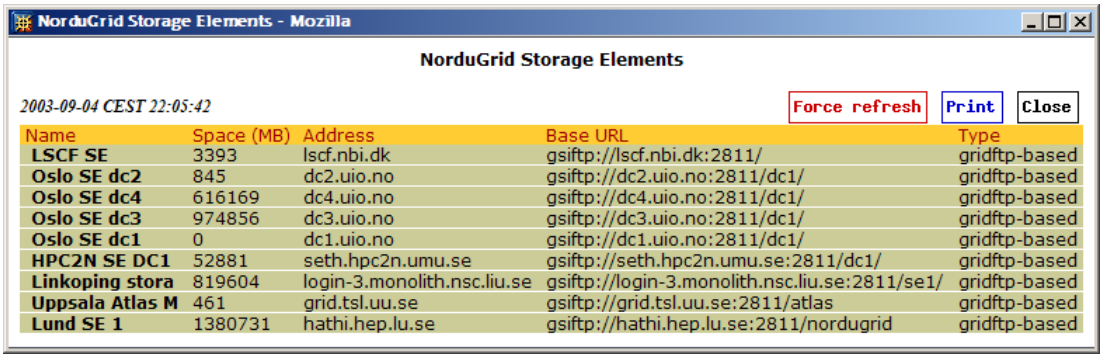

Figure 10.11: List of storage elements

10.9 List of Users

The List of Users module is different from the rest of the Grid Monitor modules because it does not deal with the NorduGrid MDS. Instead, it retrieves lists of users from the NorduGrid VO database [20]. It serves as a link between the two databases (MDS and VO), by interfacing each user record to the User Information module (Section 10.5). Figure 10.12 shows a screenshot of a typical VO user list, with numbered tags indicating clickable links as follows:

1) Name: user name as given in the corresponding Grid certificate field, linked to the User Information module (Section 10.5).

2) E-mail: E-mail address of a user, if available. It is linked to an e-mail URL, allowing a message to be sent to a user directly from the browser (if such an option is enabled in a browser).

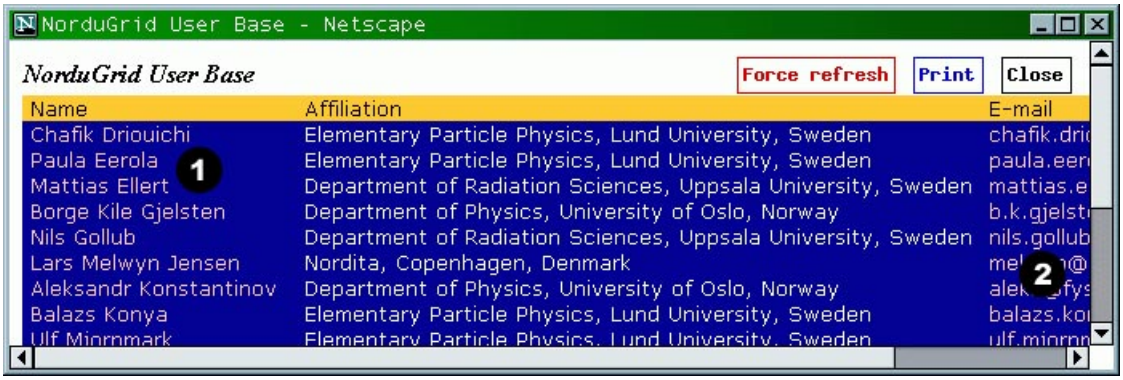

Figure 10.12: List of the NorduGrid users

The List of Users is available only from the top Grid Monitor window.

Appendix A

GACL

Many parts of ARC middleware use the Grid ACL manipulation library (GACL) to allow user control access to various objects. The GACL is a part of GridSite project [19]. To get more information about GACL please refer to the project's Web site at `http://www.gridsite.org`.

The GACL is an XML document. In ARC there can be no more that one document assigned to object (if supported). Each GACL document consist of multiple `<entry>` tags. Each such tag contains multiple and various identification tags followed by permission tags `<allow>` and `<deny>`.

Each permission tag lists applied permissions. Available values are: `<write/>`, `<read/>`, `<list/>`, `<admin/>`. Their exact meaning varies a bit depending on kind of object applied and permissions specified in `<deny>` overwrite those in `<allow>`.

Identification tags supported in ARC are as follows:

`<any-user/>`	Matches anyone.
`<auth-user/>`	User with subject name in certificate (same as `<any-user/>` in ARC.
`<person>`	Matches certificate with specified subject name.
`<dn>`	
`subject`	
`</dn>`	
`</person>`	
`<dns>`	Matches if user connects from a computer with specified hostname.
`<hostname>`	
`hostname`	
`</hostname>`	
`</dns>`	
`<voms>`	Matches if user presents credentials with matching VOMS information.
`<voms>`	All items inside `<voms>` tag are optional.
`VOMS server subject`	
`</voms>`	
`<vo>`	
`name of VO`	
`</vo>`	
`<group>`	
`VOMS group`	
`</group>`	
`<role>`	

VOMS group </role> <capability> VOMS group </capability> </voms>	
<vo>	Matches if server recognises that user belongs to specified VO.
<name> name of VO </name>	This tag is specific to server setup.
</vo>	

Here is an example of valid VOMS GACL document:

```
<?xml version=""1.0""?>
<gacl version=""0.0.1"">
 <entry>
  <voms>
   <vo>NorduGrid</vo>
   <group>developers</group>
   <role>coordinator</role>
  <voms>
  <allow><admin/><write/></allow>
 </entry>
 <entry>
  <voms>
   <vo>NorduGrid</vo>
   <group>developers</group>
  <voms>
  <allow><read/><list/></allow>
 </entry>
</gacl>
```

Appendix B

RPM For Everybody

You've been told that working with RPMs needs system administrator privileges? You've been misled.

RPM stands for *"RedHat Package Manager"* [21] and is a sophisticated archiving format, allowing not only to pack a set of files and directories in a single archive, but also to install them in an easy-to-manage manner. While a Tape Archive (`.tar`) or a ZIP archive, with which you may be familiar with, simply packs files with their relative paths, RPM stores entire directory information starting from the root. RPM also relies on a system-wide database to register software version numbers, dependencies and other relevant data. Naturally, directory structures are different on different systems, and not every user has proper permissions to modify them. Also, not everybody is allowed to write into the main RPM database. Evidently, RPM format was created having system administrators in mind. Nevertheless, even if you are a regular user with no special privileges, you can make a fair use of the RPM packages, especially so if creators of such packages made them *relocatable*, i.e., user-friendly. Luckily, most NorduGrid packages are relocatable, including the distributed Globus®RPMs. Below are some quick instructions on how to work with RPM packages without having system administrator privileges.

B.1 Listing Contents of an RPM Package

To list files archived in an RPM package, issue an RPM query command:

```
rpm -qlp myfile.rpm
```

Here command line parameters have the following meaning:

- **q** – perform a query
- **l** – list files
- **p** – use the specified package file (`myfile.rpm` in this example)

B.2 Printing Out an PM Package Information

To retrieve general information about a given RPM package, issue the following RPM query:

```
rpm -qip myfile.rpm
```

Here command line parameters have the following meaning:

- **q** – perform a query
- **i** – print out package information
- **p** – use the specified package file (`myfile.rpm` in this example)

This query will print out all the essential information stored by the creator of the RPM package in question. This may contain such fields as package name and version, creator's name, build date and – what's most important for users – ***relocations***. Section B.7 explains how to make use of this information.

B.3 Simple Unpacking of an RPM Archive

In many cases you would only need to extract files from an RPM archive, without performing complex operations with an RPM database. There is a simple way to do it:

```
rpm2cpio < myfile.rpm | cpio -i --make-directories
```

This method makes use of the generic GNU CPIO utility, which deals with copying files to and from archives. The `rpm2cpio` command converts from RPM to CPIO format, and the output of this conversion is redirected to the `cpio` command. Command line parameters for the latter have the following meaning:

i	–	extract contents
make-directories	–	create directories when necessary

As a result, all the contents of the RPM package will be extracted into the current directory, creating relative paths.

B.4 Creating a Private RPM Database

To achieve higher flexibility and to get access to more RPM functionality, you may want to make a step forward and create your own RPM database. You only have to do it once, by issuing the following instructions:

```
mkdir $HOME/myrpmdb
rpmdb --initdb --dbpath $HOME/myrpmdb
```

Mentioned in this example directory `$HOME/myrpmdb` can of course be substituted with any other location which suits you best. The only requirement is that you must have write permissions in this location.

Having such a personal RPM database allows you to perform more complex operations with RPM packages, without resorting to simple unpacking and manual installation. To simplify your life even further, do the following:

```
echo '%_dbpath /home/yourname/myrpmdb' | cat >> $HOME/.rpmmacros
```

This will create a file `.rpmmacros` in your `$HOME` directory*, and add there the line specifying the database location. In this example, `/home/yourname/myrpmdb` should be substituted with your RPM database path. This will instruct RPM to use always your personal database. If you don't want to use this feature (for example, if you have several databases), take care to append

```
--dbpath $HOME/myrpmdb
```

to each RPM instruction which refers to such a database (e.g., package installation or removal) .

It may be useful to copy the system-wide database into your own one. RPM does not provide any tool for database synchronization (yet?), thus you have to do it hard way:

```
cp /var/lib/rpm/* $HOME/myrpmdb/.
```

Here `/var/lib/rpm/` is a typical default location of the system-wide database (it could be different though on different systems). Having thus your private database initialized with the existing system information may help you in dealing with *dependencies* (see Section B.8), but only for a while, since any system-wide upgrades will not be registered in your database, and there is no way to synchronize RPM databases in a clever manner.

*On some systems, you should use > instead of >> if the file `.rpmmmacros` did not exist before.

B.5 Installing an RPM Package

Having defined your private database (Section B.4), you can attempt to install any RPM package by doing the following:

```
rpm -ivh myfile.rpm
```

Here command line parameters have the following meaning:

- **i** – perform an installation procedure
- **v** – print extra information
- **h** – print progress bar with hash-marks

Keep in mind that if you didn't define a default database in `$HOME/.rpmmacros` , you'll have to append `--dbpath $HOME/myrpmdb` (or whatever is your private database location) to the directive.

Most likely, however, such a simple installation instruction will fail. There are two main reasons:

a) RPM packages attempt to install themselves in locations defined by their creators, not by you. How to get around this is described in Section B.7.

b) RPM checks for other software needed by the package – *dependencies* – in your private database; however, most likely, such software was not installed by you and can not be found in your database. How to deal with this is described in Section B.8.

B.6 Upgrading RPM Packages

If you happened to have a package installed and simply want to upgrade it, issue the following instruction:

```
rpm -Uvh myfile.rpm
```

Here command line parameters have the following meaning:

- **U** – upgrade the installation
- **v** – print extra information
- **h** – print progress bar with hash-marks

Since this operation needs to access a database, you either have to have the default database location specified in the `$HOME/.rpmmacros` file, or to append `--dbpath $HOME/myrpmdb` (or whatever is your private database location) to the directive (see Section B.4 for details).

B.7 Relocating RPM Packages

As it was mentioned in Section B.2, an RPM package may contain information on relocatable directories. Such directories are listed as *relocations*, meaning that their contents can be relocated during the installation in another directory. To perform such an installation, do the following:

```
rpm -ivh myfile.rpm --relocate /oldlocation=/newlocation
```

Here `/oldlocation` is one of the relocatable directories of the original package as listed by the `rpm -qip myfile.rpm` command, and `/newlocation` is your preferred destination, specified as an absolute path (starting with "`/`"). The command line parameters have the following meaning:

- **i** – perform an installation procedure
- **v** – print extra information
- **h** – print progress bar with hash-marks
- **relocate** – instruct RPM to install files from `/oldlocation` into `/newlocation`

As of RPM version 4.2.1, multiple relocations are possible[†]. To use this option, specify several relocation pairs on the command line, i.e.,

```
--relocate /old=/new --relocate /old/subdir=/new/subdir.
```

The order of such pairs in the line is important, as RPM will attempt to create new directories and, quite naturally, can not skip a level.

If the package information (as listed by `rpm -qip`) contains several relocations, you must specify a `--relocate` pair for **a**ll of them.

It may turn out that the package creator forgot to describe some pathes as "relocateable", so that they do not appear in the relocations list of the RPM, and yet the package attempt to install in such directory. There's little you can do about it, except attempting to use the `--badreloc` option:

```
rpm -ivh myfile.rpm --relocate /oldlocation=/newlocation --badreloc
```

As in the case of installation, you either have to have the default database location specified in the `$HOME/.rpmmacros` or to append `--dbpath $HOME/myrpmdb` private database location) to the directive (see Section B.4 for details).

B.8 Dealing with Dependencies

Even when relocations proceeded smoothly, your package may still fail to get installed due to unsatisfied *dependencies* (this is the name for any other software needed by the package), which are absent from your private database (Section B.4). A bold way to deal with it is to force the installation, e.g.:

```
rpm -ivh myfile.rpm --relocate /oldlocation=/newlocation --nodeps
```

The command line parameters have the following meaning:

i – perform an installation procedure
v – print extra information
h – print progress bar with hash-marks
relocate – instruct RPM to install files from `/oldlocation` into `/newlocation`
nodeps – ignore dependencies

This is an admittedly non-elegant way. A better solution may be to enter the necessary information into the database by installing "virtual" RPMs which only register a database record without actually installing software; however this is clearly package-dependent, thus you should request such a "fake" RPM from the package provider.

B.9 Listing Installed Packages

To list all the packages installed via an RPM mecanisms, do:

```
rpm -qai
```

Here command line parameters have the following meaning:

q – perform a query
a – query all installed packages
i – print out package information

[†]Some older versions can also do multiple relocations, but most have one or another bug.

This command prints out the information stored in your RPM database, hence you either have to have the default database location specified in the `$HOME/.rpmmacros` file, or to append `--dbpath $HOME/myrpmdb` (or whatever is your private database location) to the directive (see Section B.4 for details).

B.10 Uninstalling RPM Packages

Whenever you'll have to uninstall a package, issue the instruction:

```
rpm -e mypackage
```

Here `mypackage` is the name of the package as listed by the `rpm -qai` command. Command line parameter has the following meaning:

e – uninstall packages

This command removes the files associated with the package from your system and erases the corresponding record from your RPM database. You either have to have the default database location specified in the `$HOME/.rpmmacros` file, or to append `--dbpath $HOME/myrpmdb` (or whatever is your private database location) to the directive (see Section B.4 for details).

B.11 Building RPM from Source

It may happen so that binary RPM packages for your system are unavailable, but a source RPM or a tarball is provided. You can build a binary RPM yourself from such sources, but some extra manipulations should be made first.

The default RPM top directory is `/usr/src/redhat` (for a RedHat system). Naturally, as a user, you have no write permission there. To solve the problem, start by creating a new directory structure in any place which has enough disk space and write permission:

```
mkdir -p rpmtop/RPMS/i386; mkdir rpmtop/SRPMS; mkdir rpmtop/SOURCES;
mkdir rpmtop/BUILD; mkdir rpmtop/SPEC
```

Now you should instruct RPM to use this directory instead of the system-wide one (assuming you created the `rpmtop` structure above in `/home/yourname/`):

```
echo '%_topdir /home/yourname/rpmtop' | cat >> $HOME/.rpmmacros
```

Next thing you should take care of, is the temporary directory for RPM builds, which by default is `/var/tmp/`, also unavailable for mere mortals. The solution is similar to the above, to create your own directory in any place which has enough disk space and write permission:

```
mkdir mytmp
```

This directory should also be specified in your `.rpmmacros` file:

```
echo '%_tmppath /home/yourname/mytmp' | cat >> $HOME/.rpmmacros
```

That's it; from now on you should be able to build binary RPMs from source by using the command:

```
rpmbuild -ba mypackage.src.rpm
```

or

```
rpmbuild -ta mypackage.tar.gz
```

depending whether the sources are provided as source RPM (`mypackage.src.rpm`) or as a tarball (`mypackage.tar.gz`). The command line options are, correspondingly:

ba – build binary and source packages from a source RPM

ta – build binary and source packages from (gzipped) tarball

Your RPMs will be placed in `/home/yourname/rpmtop/RPMS/i386/` directory of the example above. In case your architecture is different from `i386`, please take care of creating a corresponding subdirectory in `rpmtop/RPMS` (e.g., `rpmtop/RPMS/alpha`). If you do not know what is the architecture, use

```
uname -m
```

This command will typically print out a string like "i386" or "i586" or "i686" and such, which is the hardware architecture name of your machine.

Appendix C

Globus Toolkit installation notes

ARC requires Globus Toolkit 4 or 5. You can get Globus Toolkit 5 from NorduGrid repositories for the supported Linux distributions. If you need special installation instructions, please refer to the Globus installation guide.

Appendix D

NorduGrid ARC build notes

This section describes the build procedure of the NorduGrid middleware (ARC) from tarball or from source RPM.

D.1 Dependencies

Before trying to compile the ARC tools, make sure you have installed the following software packages:

- GNU `make`, GNU `autotools`
- C++ compiler and library
- Grid Packaging Tools (GPT) [22]
- Globus Toolkit 4 or 5 [2] which contains
  - GridFTP client libraries
  - Globus RLS
  - RSL libraries
- `gSOAP` [23]
- `GACL` [19]
- `VOMS` [24]
- `MySQL` client libraries
- `libxml2`

Note that NorduGrid provides packaged versions of GPT, Globus, `gSOAP` and `VOMS`, while `GACL` is distributed within the NorduGrid source. The rest of the dependencies are available in any standard Linux distributions.

D.2 Basic build with autotools

The build is using the GNU `autotools`. This means that a standard installation can be made with:

```
./configure
make
make install
```

The `configure` script accepts the usual options with the addition of:

`--enable-experimental`	Enable experimental code
`--enable-rpath-hack`	Do not rpath Globus libraries in binaries
`--enable-docs`	Build the documentation
`--with sysv-scripts-location=<PATH>`	Location of the SYSV init scripts (e.g. `/etc/init.d`)
`--with-monitor-prefix=<PATH>`	Specify the location of the Grid Monitor PHP scripts
`--with-monitor-local-prefix=<PATH>`	Specify the relative location of the Grid Monitor PHP scripts
`--with-gsoap-location=<PATH>`	Specify the `gSOAP` installation path
`--with-mysql-location=<PATH>`	Specify the `MySQL` installation path
`--with-libxml-location=<PATH>`	Specify the `libxml` installation path
`--with-nordugrid-location=<PATH>`	Specify the ARC installation path
`--with-globus-location=<PATH>`	Specify the Globus installation path
`--with-gpt-location=<PATH>`	Specify the GPT installation path
`--with-globus-makefile-header`	Use `globus-makefile-header` to generate `makefile` stubs
`--with-flavor=<flavor>`	Specify the Globus build flavor or `without-flavor` for a flavor independent build

Note that using `--with-nordugrid-location=/opt/nordugrid` switch will install the software using the `/opt/nordugrid` prefix **and** place the software in subdirectories:

```
/opt/nordugrid/bin
/opt/nordugrid/lib
/opt/nordugrid/libexec
..
```

For a "standard" install the `--prefix=/usr/local` will place the software in nordugrid subdirectories:

```
/usr/local/bin
/usr/local/lib/nordugrid
/usr/local/libexec/nordugrid
..
```

Please note that as of the 0.4 Release the "standard install" is not fully supported yet, we recommend to use the `--with-nordugrid-location` switch. Furthermore, `make` builds the full middleware containing server, client, monitor and documentation; modular builds (such as `make client`, or `make server`) are not supported yet.

D.3 Building with RPM

NorduGrid can be built using RPM. From a `tar.gz` file, do:

```
rpmbuild -ta nordugrid-arc-<version>.tar.gz
```

Or, from a `.src.rpm` do:

```
rpmbuild --rebuild nordugrid-arc-<version>.src.rpm
```

Source RPMs are configured with the `--with-nordugrid-location` switch and binary builds are fully relocatable.

In terms of RPMs the build requirements can be satisfied with the following packages:

- `gpt >= 3`
- `globus >= 2.4.3-9ng`
- `gsoap >= 2.5.2`
- `voms = 1.9.14`
- `voms-devel = 1.9.14`
- `libxml2-devel`
- `MySQL-devel`

Note that the naming convention above might not correspond exactly to the naming used on your system. Also be aware that on some systems the RPM dependencies are broken. This means for example that `libxml2` may not be explicitly required by `lixml2-devel` while it should be.

D.4 Binary RPMs: runtime requirements

The RPMs created by `rpmbuild` as in Section D.3 are:

- `nordugrid-arc-client` – Client programs
- `nordugrid-arc-server` – Server-side meta package depending on all necessary packages
- `nordugrid-arc-gridftpd` – GridFTP package
- `nordugrid-arc-grid-manager` – Grid-manager package
- `nordugrid-arc-infosys-ldap` – Infosys and infoindex package
- `nordugrid-arc-gridmap-utils` – Utilities for managing gridmap-files
- `nordugrid-arc-ca-utils` – Utilities for maintaining CA files
- `nordugrid-arc-devel` – Development files (headers, static libs)
- `nordugrid-arc-monitor` – Grid monitor web interface (should be installed on a Web server)
- `nordugrid-arc-doc` – Documentation

The strict runtime RPM dependencies are:

- `nordugrid-arc-client -> globus`
- `nordugrid-arc-server -> globus, globus-config, voms = 1.9.14, libxml2`
- `nordugrid-arc-gridmap-utils -> perl-perl-ldap, perl-libwww-perl`
- `nordugrid-arc-ca-utils -> wget`

Dependencies not expressed in the RPM requirements:

- `nordugrid-arc-server ->` standard Perl environment
- `nordugrid-arc-ca-utils -> openssl` (native or globus-provided)
- `nordugrid-arc-gridmap-utils -> openssl` (native or provided with Globus) and optional (preferably) `curl >= 7.9.8`

Special requirements of the Grid Monitor (not expressed in RPM requirements are described in the README file of the `nordugrid-arc-monitor` RPM.

Appendix E

Known Grid Certificate Authorities

This list of Certificate Authorities is non-authoritative, as the authorities and links are subject to change.

Country	CA	URL
Albania	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
Armenia	ArmeSFo	http://www.escience.am/ca
Austria	AustrianGrid	http://ca.austriangridca.at
Belgium	BEGrid	https://gridra.belnet.be/pub
Bosnia and Herzegovina	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
Bulgaria	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
Canada	GridCanada	http://www.gridcanada.ca/ca
China	IHEP Grid	https://gridca.ihep.ac.cn
Croatia	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
Cyprus	CyGrid	http://www.cs.ucy.ac.cy/cygrid-ca
Czech Republic	CESNET	http://www.cesnet.cz/pki
Denmark	NorduGrid	http://www.nbi.dk/HEP/CA
Estonia	Eesti Grid	http://grid.eenet.ee/en
Finland	NorduGrid	http://www.nbi.dk/HEP/CA
France	CNRS	http://igc.services.cnrs.fr/Datagrid-fr
Germany	GermanGrid	http://grid.fzk.de/ca
	DFN-GridGermany	http://www.pca.dfn.de
Greece	Hellas Grid	http://pki.physics.auth.gr/hellasgrid-ca
Hungary	NIIF	http://www.ca.niif.hu
	KFKI RMKI	http://pki.kfki.hu
	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
Ireland, Republic of	Grid-Ireland	http://www.cs.tcd.ie/grid-ireland/gi-ca
Israel	IUCC	http://certificate.iucc.ac.il
Italy	INFN	http://security.fi.infn.it/CA
Korea, Republic Of	KISTI	http://gridtest.gridcenter.or.kr
Macedonia	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
The Netherlands	NIKHEF	http://certificate.nikhef.nl
Norway	NorduGrid	http://www.nbi.dk/HEP/CA
Pakistan	PK-Grid	http://www.ncp.edu.pk/pk-grid-ca
Poland	Polish Grid	http://www.man.poznan.pl/plgrid-ca

Portugal	LIP	http://ca.lip.pt
Romania	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
Russia	RDIG	http://lhc.sinp.msu.ru/CA
Serbia and Montenegro	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
Slovakia	SlovakGrid	http://ups.savba.sk/ca
Slovenia	SiGNET	http://signet-ca.ijs.si
Spain	Datagrid-ES	http://www.ifca.unican.es/datagrid/ca
Sweden	NorduGrid	http://www.nbi.dk/HEP/CA
Switzerland	SWITCH	http://www.switch.ch/pki
	CERN	http://cern.ch/service-grid-ca
Taiwan	ASCCG	http://ca.grid.sinica.edu.tw
Turkey	SEE-Grid	http://www.grid.auth.gr/pki/seegrid-ca
United Kingdom	e-Science Grid	http://www.grid-support.ac.uk/ca
United States	DOE Science Grid	http://www.doegrids.org
	ESnet	http://www.es.net/CA
	FNAL	http://computing.fnal.gov/security/pki

Bibliography

[1] "The NorduGrid Collaboration," Web site. [Online]. Available: http://www.nordugrid.org

[2] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer Applications*, vol. 11, no. 2, pp. 115–128, 1997, available at: http://www.globus.org.

[3] A. Konstantinov, *The NorduGrid Grid Manager And GridFTP Server: Description And Administrator's Manual*, The NorduGrid Collaboration, NORDUGRID-TECH-2. [Online]. Available: http://www.nordugrid.org/documents/GM.pdf

[4] B. Kónya, *The NorduGrid/ARC Information System*, The NorduGrid Collaboration, NORDUGRID-TECH-4. [Online]. Available: http://www.nordugrid.org/documents/arc_infosys.pdf

[5] M. Ellert, *ARC User Interface*, The NorduGrid Collaboration, NORDUGRID-MANUAL-1. [Online]. Available: http://www.nordugrid.org/documents/ui.pdf

[6] O. Smirnova, *Extended Resource Specification Language*, The NorduGrid Collaboration, NORDUGRID-MANUAL-4. [Online]. Available: http://www.nordugrid.org/documents/xrsl.pdf

[7] ——, *The Grid Monitor*, The NorduGrid Collaboration, NORDUGRID-MANUAL-5. [Online]. Available: http://www.nordugrid.org/documents/monitor.pdf

[8] A. Sim, A. Shoshani, *et al.*, "The Storage Resource Manager Interface (SRM) Specification v2.2," May 2008, GFD-R-P.129. [Online]. Available: http://www.ggf.org/documents/GFD.129.pdf

[9] RSA Labs, "RSA Cryptography FAQ." [Online]. Available: http://www.nordugrid.org/documents/rsalabs_faq41.pdf

[10] "Public-Key Infrastructure (X.509) (PKI), Proxy Certificate Profile." [Online]. Available: http://rfc.net/rfc3820.html

[11] "The Open Source toolkit for SSL/TLS," Web site. [Online]. Available: http://www.openssl.org/

[12] M. Smith and T. A. Howes, *LDAP : Programming Directory-Enabled Applications with Lightweigt Directory Access Protocol.* Macmillan, 1997.

[13] I. Foster *et al.*, "A Security Architecture for Computational Grids," in *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security.* ACM Press, November 1998, pp. 83–92.

[14] "The Globus Resource Specification Language RSL v1.0." [Online]. Available: http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html

[15] A. Konstantinov, *The NorduGrid Smart Storage Element*, The NorduGrid Collaboration, NORDUGRID-TECH-10. [Online]. Available: http://www.nordugrid.org/documents/SE.pdf

[16] "gLite, Lightweight Middleware for Grid Computing," Web site. [Online]. Available: http://glite.web.cern.ch/glite/

[17] H. Stockinger *et al.*, "File and Object Replication in Data Grids," *Cluster Computing*, vol. 5, no. 3, pp. 305–314, July 2002.

[18] A. L. Chervenak *et al.*, "Performance and Scalability of a Replica Location Service," in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04).* IEEE Computer Society Press, 2004, pp. 182–191.

[19] A. McNab, "The GridSite Web/Grid security system: Research Articles," *Softw. Pract. Exper.*, vol. 35, no. 9, pp. 827–834, 2005.

[20] "Description of the NorduGrid Virtual Organisation." [Online]. Available: http://www.nordugrid.org/NorduGridVO/vo-description.html

[21] "RedHat Package Manager," Web site. [Online]. Available: http://www.rpm.org

[22] "Grid Packaging Tools," Web site. [Online]. Available: http://www.gridpackagingtools.org

[23] R. A. van Engelen, "gSOAP." [Online]. Available: http://www.cs.fsu.edu/~engelen/soap.html

[24] R. Alfieri *et al.*, "From gridmap-file to VOMS: managing authorization in a Grid environment," *Future Gener. Comput. Syst.*, vol. 21, no. 4, pp. 549–558, 2005.

Index