



The VBrowser

The VBrowser Installation and User's Guide

(For version 1.6)

`vlet-develop (at) lists.sourceforge.net`
`http://www.vl-e.nl/vbrowser`

netherlands



center

Netherlands eScience Center

Science Park 140 (Matrix I)
1098 XG Amsterdam
The Netherlands

Information:

Mail: [vlet-develop \(at\) lists.sourceforge.net](mailto:vlet-develop@lists.sourceforge.net)
Web: <http://github.com/NLeSC/vbrowser/>

Contents

1	Introduction	6
2	Installation	7
2.1	Installation requirements	7
2.2	Unpacking	7
2.2.1	Quick Installation Steps	8
2.2.2	Directory structure	8
2.2.3	Configuring vbrowser.ini for Windows	9
2.2.4	Configuring the installation	10
2.3	Configuring Grid Certificates	10
2.3.1	Configuring your personal Grid certificate	10
2.3.2	Configuring host or “root CA” certificates	11
2.3.3	Advanced certificate configuration settings	12
2.4	Properties and configuration settings	12
2.4.1	Firewall settings	12
2.4.2	Grid proxy and grid certificate locations	12
2.4.3	Command line options and environment variables	13
3	Using the VBrowser	14
3.1	Starting the VBrowser	14
3.1.1	VBrowser panel overview	14
3.1.2	Location and navigation bar	15
3.1.3	Toolbars	16
3.1.4	Table panel	16

3.1.5	Pop-up or Action menu	18
3.2	Authenticating yourself with the Grid	18
3.3	Grid Neighbourhood	19
3.4	Configuring your personal environment	20
3.4.1	Adding Resources	20
3.4.2	Adding New (Grid) Resource Location	22
3.4.3	Adding an LFC Location	23
3.5	Links or shortcuts	28
4	Other Tools	29
4.1	Application and Tools	29
4.2	GUI utils	29
4.2.1	GridProxyDialog	29
4.2.2	VLTerm	29
4.3	Command line tools	31
4.3.1	URI script interface	31
4.3.2	Examples	32
4.3.3	Customized uricopy script	33
4.4	Jython	34
4.4.1	Jython start script: jython.sh	34
5	Customization	39
5.1	Custom viewers/plugins	39
5.2	Custom Mimetypes and icons	40
5.2.1	User defined mimetypes	40
5.2.2	Web server mimetypes	41
5.2.3	Magic types	42
5.2.4	Custom icons	42
5.2.5	Viewer preferences and defaults	43
A	Appendices	45
A.1	VRL specification	45

A.1.1	Syntax	46
A.1.2	Examples of URIs	46
A.1.3	URI Attributes	47
A.2	Overview of configuration files and directories	48
A.2.1	Installation files and settings	48
A.2.2	Default installation configuration file <code>vletrc.prop</code>	48
A.2.3	User configuration files	49
A.2.4	Example user customized <code>mime.types</code> file	49
A.2.5	Example user customized <code>viewerconf.prop</code> file	49

Chapter 1

Introduction

The VBrowser Toolkit.

This is the Installation and User Guide to the **VBrowser** toolkit. This document introduces you to the VBrowser environment and is divided into the following parts:

- Part I: Introduction & Installation: Chapter 1 and 2.
- Part II: Using the VBrowser: Chapter 3 and 4.
- Part III: Advanced features and customizing settings: Chapter 5.
- Appendices.

Please report any problems, bugs to: `vlet-develop` (at) `lists.sourceforge.net` with the subject: 'bug report' (This is a moderated list, so you may receive a notification that your message is pending 'approval').

For more information see the GitHub site at <https://github.com/NLeSC/vbrowser/wiki>.

Chapter 2

Installation

2.1 Installation requirements

The VBrowser has the following minimal installation requirements:

- SUN Java 1.6 JRE (Java Runtime Environment) or JDK is recommended.
- Either Windows 2000/XP/7 or a recent Linux distribution (2.6 kernel)
- Other Java 1.6 capable platforms might work as well, but not all architectures have been tested. (Mac Os should work as well).
Alternative Java implementations like 'gcj' and 'openjdk' might not fully support Swing applications which is mandatory for the VBrowser.

Also, to be able to access grid resources, you MUST have a grid certificate. For instructions how to get one, see: <http://certificate.nikhef.nl/request> or ask your local grid administrator.

2.2 Unpacking

Unzip the package with winzip (windows) or gunzip (linux). The installation directory into which you installed it, will be referred to as VLET_INSTALL.

Default installation path on Linux (system wide) installation could be:

`/opt/vlet`

The installation path on Linux for non system administrators, could be:

`\$HOME/vlet`

Default installation path for a windows (system wide) installation could be:

`C:\vlet`

As users do not need write access to the installation path nor administrator rights to install or run parts of the VL-e Toolkit, any user can install it at the desired place he or she wants to.

After unpacking the VL-e toolkit can be used directly by starting the desired application or tool from the `VLET_INSTALL/bin` directory. Most users don't have to configure their installation and should be able to use the tools right out of the box. For custom configurations, see the next chapters for an explanation of optional properties and installation settings.

2.2.1 Quick Installation Steps

Quick install steps for the impatient are as follows:

- Unzip the vlet distribution.
- Put your personal grid certificate (`userkey.pem` and `usercert.pem`) in your `HOME/.globus` directory. Create the `.globus` directory if it doesn't exist yet.
- Optionally create a shortcut to `VLET_INSTALL/bin/vbrowser.exe` (for **Windows**) and copy the shortcut to your desktop.
- Start `vbrowser.sh` or `vbrowser.exe` from `VLET_INSTALL/bin` (or use shortcut).
- Optionally, the `vbrowser.jar` can be used to directly start the `vbrowser` using any Java Environment (for Mac OS this is the only supported way) as follows: `java -jar vbrowser.jar`
- Configure firewall settings by right-clicking (alt mouse button) on the **MyVLe** icon seen in Figure 2.1, select **properties** and set `passiveMode` to `true` for firewalled users, or specify an incoming port range if you allow incoming tcp/ip connections.



Figure 2.1: The MyVLe Icon

- Optionally put extra CA certificates in your `HOME/.vletrc/certificates` directory if CA certificates can't be found. This is necessary for non-DUTCHGRID sites.

2.2.2 Directory structure

Below an overview of the directory structure of the installation. Only relevant files and directories are listed here.


```
# Directory structure of VLET distribution:

ReleaseNotes.txt      # latest release notes.
README.txt            # general README.
INSTALL.txt           # extra installation notes.
ReleaseNotes.txt      # latest release notes.
LICENCE.txt           # Apache Licence 2.0.
bin/                  # scripts and (binary) executables.
bin/vbrowser.sh        # UNIX (Linux) VBrowser startup script.
bin/vbrowser.exe       # Windows VBrowser startup executable.
bin/vbrowser.jar       # Java executable jar file.
etc/                  # configuration files.
etc/vletrc.prop        # installation settings.
etc/vletenv.sh         # extra Unix/Linux environment settings.
etc/certificates/      # root CA certificates needed for authentication of
                        # remote hosts.
doc/                  # documentation and Java API (javadoc) files.
lib/                  # libraries and needed jar files including icons.
lib/plugins            # custom viewers and plugins both for VDrivers and
                        # Viewer Plugins.
lib/icons              # custom icons.
lib/icons/mimetypes    # custom mimetype icons.
py/                   # Java Python (Jython) examples.
```

2.2.3 Configuring vbrowser.ini for Windows

Under Windows, the file 'vbrowser.exe' (located in the 'bin' directory) starts the VBrowser. This executable uses the default Java version installed and starts the Java executable jar file 'vbrowser.jar'. If you want to specify an alternative java installation, or the default java installation can't be found, you can specify it in a configuration file 'vbrowser.ini' located in the same directory as 'vbrowser.exe'. If on windows VLET is installed in for example C:\vlet\ you can create the file **vbrowser.ini** at the following path:

```
C:\vlet\bin\vbrowser.ini
```

In that file you can add the following lines:

```
##
# Use following Java installation:
java.home=C:\Program Files\Java\jdk1.7
```

You don't have to put the path between quotes, but make sure no extra spaces are after the last part of the path (..\jdk1.7).

Also you can specify extra Java VM options, for example to increase the Java memory settings:

```
# Extra Java VM options. Increase heap size:
java.vargs=-Xms256m -Xmx512m
```

2.2.4 Configuring the installation

The next sections explain how to configure the default settings in your installation. Most settings can be changed interactively through the **VBrowser** as well. You can do this by right-clicking (or use alt-mouse button) on the **MyVLe** icon or specified resource.

See the file `VLET_INSTALL/etc/vletrc.prop` for (advanced) installation configuration. All the variables can also be specified for each user in the `HOME/.vletrc/vletrc.prop` which overrides default installation settings. (see Section: [2.3.1](#)). As this file changes per distribution, see the comments before each property and which are the optimal defaults for your distribution and your environment.

2.3 Configuring Grid Certificates

2.3.1 Configuring your personal Grid certificate

The next paragraphs explain where you can find/store your grid certificate depending on the operating system you are using.

Grid certificate directory for Linux

The default location to store your grid certificate for **Linux** is:

```
HOME/.globus
```

This is the location where the Globus Commodity Grid Toolkit (CoG) stores its configuration files. This location is the recommended place for Linux and windows users to store their grid certificates (See next section how to find your windows home).

Grid certificate directory for Windows

The default location for **Windows** (XP) could be (depending on the default user's home):

```
C:\WINDOWS\profiles\Your User Name\globus
```

Or on Windows 7, this could be:

```
C:\Users\Your User Name\globus
```

The `C:\WINDOWS` part is the prefix where windows is installed. The `.globus` is the directory where the Globus Commodity Grid Toolkit (CoG) stores its configuration files.

If you don't know how to find your Windows **HOME** you can also just start the **VBrowser** and your home path will be the first resource under the **Local System** icon. This home resource is recognizable by the folder icon which has a little house in it as you can see in [Figure 2.2](#).

Alternatively you could store your certificates into another location, or keep a copy, for example the following location:

```
C:\My Documents\globus
```

When installing your grid certificates into another location than the default, you need to configure the **VBrowser** to search for this location. This will be explained in [Section 2.4.2](#).

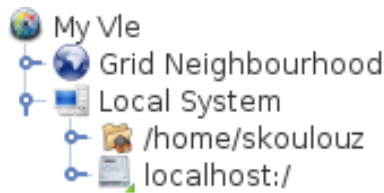


Figure 2.2: Your HOME folder under window and Linux

Your Grid Certificate files

The files which must be present in your globus certificate directory (`HOME/.globus`) are:

- `userkey.pem` : Your *private* grid certificate. Never share this file with anyone. If you loose this file, you'll need to reapply for a new grid certificate.
- `usercert.pem` : Your *public* grid certificate. This file is send over the internet to identify you as the person defined in your grid certificate. Together with the `userkey.pem` it forms your *public-private* keypair.

For more about *public-private* keypair encryption, see: [8].

Optionally this directory may contain the following files:

- `cog.properties` : This file contains the Globus Commodity Grid (CoG) Toolkit properties.

Note: The `cog.properties` file is *always* stored in the default globus configuration directory (`.globus`) in your HOME directory. If this file is present it will override any configurations made in `VLET_INSTALL/etc/vletrc.prop` or `HOME/.vletrc/vletrc.prop`

2.3.2 Configuring host or “root CA” certificates

The default place to store host certificates under **Linux** is:

`/etc/grid-security/certificates`

If you don't have the rights to add/install certificates to this place, you can add new certificates to the following location:

`VLET_INSTALL/etc/certificates`

Or to add root CA certificates to your user environment put them in:

`HOME/.vletrc/certificates`

Certificates added to the above locations are loaded automatically when starting tools from the VL-e toolkit. For windows users replace the `HOME` variable with your user profile location.

For access to DUTCHGRID sites, no extra configuration should be necessary. To access other (non DUTCHGRID) sites, add your root CA certificate to one of the above custom locations.

More information about Grid Certificates can be found here: [6]

2.3.3 Advanced certificate configuration settings

See the installation file `VLET_INSTALL/etc/vletrc.prop` or the user configuration file `HOME/.vletrc/vletrc.prop` for advanced grid configuration settings. Explanation of these properties can be found in [Section:2.4](#)

2.4 Properties and configuration settings

In the next (sub)sections an overview of the most important settings which can be specified either installation wide in the `VLET_INSTALL/etc/vletrc.prop` file or in the user's configuration file `HOME/.vletrc/vletrc.prop`.

2.4.1 Firewall settings

- `firewall.portrange`: Allowed incoming port range for hosts which have a 'hole' in their firewall. For example:

```
firewall.portrange=20000,25000
```

Leave empty if no range is defined.

- `passiveMode`: Set to true when incoming connections are NOT allowed (firewall port-range will be ignored!). For example:

```
passiveMode=true
```

Note: if you set `passiveMode` to true, *all* server configurations will use `passiveMode`. Keep this setting to false if you want to specify this option per server. This is the case when you have file servers which are behind the same (company) firewall and are not blocked when accessing them from your desktop.

Performance note: Allowing incoming connections can considerably speed up file transfer and is recommended for large file transfers even for low bandwidth connections and especially for connections which have a high (tcp/ip) latency.

2.4.2 Grid proxy and grid certificate locations

You can specify alternate locations to your Grid Certificate and Proxy Location. Beware that when you specify these setting in the installation configuration (`VLET_INSTALL/etc/vletrc.prop`), you'll be specifying this for *all* users.

Use `HOME/.vletrc/vletrc.prop` for personalized settings. You can copy any installation property from the installation file in to your own user property file to customize your environment.

- `grid.proxy.location`: absolute path to grid proxy location or relative from user's HOME. For example (absolute path):

`grid.proxy.location=/etc/ptdeboer_proxy.x509`

or a location relative to the user's HOME:

`grid.proxy.location=mycerts/myproxy.x509`

or (preferably) leave empty to use (CoG/globus) defaults.

- `grid.certificate.location`: Relative path (to user's home) or absolute path to the directory containing your grid certificates (both public and private keys). Leave empty to use (globus) defaults.

`grid.certificate.location=.globus`

Above setting is the default location as used by CoG/globus.

2.4.3 Command line options and environment variables

All properties can be set by specifying them as extra arguments on the command by using the `-D` command as follows: `-D<variable>=<value>`. The order in which properties are checked is as follows (higher priority first):

- Command line options `-D<variable>=<value>`
- Environment variables.
- User configuration file: `HOME/.vletrc/vletrc.prop`
- Installation configuration file: `VLET_INSTALL/etc/vletrc.prop`
- Hard-coded defaults (for configuration-less environments)

Chapter 3

Using the VBrowser

3.1 Starting the VBrowser

3.1.1 VBrowser panel overview

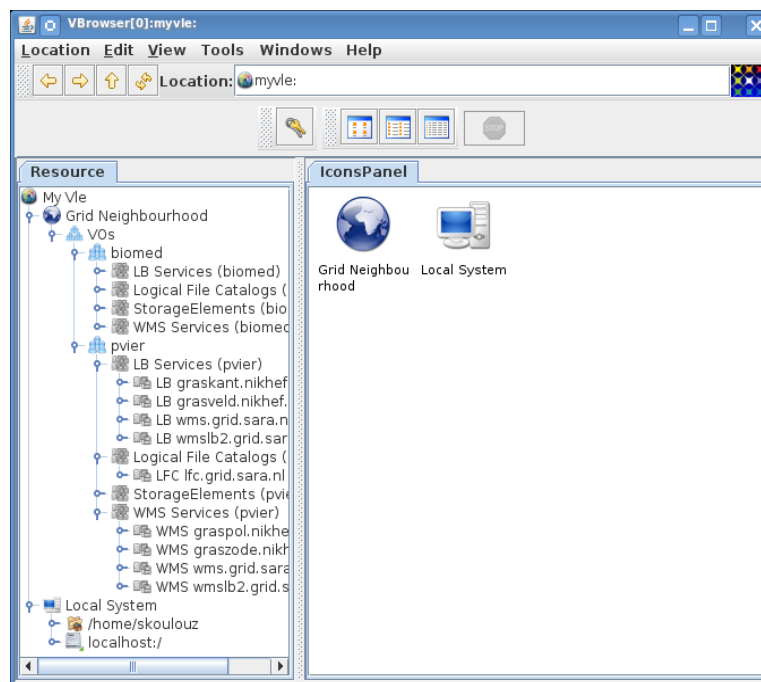


Figure 3.1: VBrowser window

Resource Tree panel

In the ResourceTree (Figure 3.1 left panel) in the upper left there is the main resource or root resource called **MyVLe** (You can rename this resource). Directly below **MyVLe** you can see the **Grid Neighbourhood** containing your VOs and the resources associated with them (see Section ??). Under that there is the **Local System** with your home directory. All the resources in this tree represent a logical ordering of your resources. When clicking on a tree node the VBrowser will connect to the remote server or resource

and 'open' the location. If the user doesn't click on the node or expands the tree the remote server or resource will not be opened and no connection is used. Some actions might require that the user first has opened the resource.

Icons panel

The icons panel is by default the panel you see on the right when starting the **VBrowser**.

Actions

Double click (Action Click) on the resources (their icons) to open these location or right click (or use alternative mouse button) to view and/or edit the properties of these resources.

You can add new resources by right-clicking (or use alt-mouse-button) on **MyVLe** and select **new**. This also works by right clicking (alt-mouse-button) on the white space between the icons or *canvas* in the IconsPanel on the right.

Menu Bar

The menu bar has six menus:

- **Location**. This includes sub-menus like New Window, Open, Close, etc.
- **Edit**, with cut, copy, paste, etc.
- **View**. This includes sub-menus like View as Icons or List, and Preferences. With the Preferences sub-menu the user can change between single and double click mouse behaviour.
- **Tools** which contain the VLTerm, a terminal for SSH and bash sessions (see Section 4.2.2) as well as some external tools.
- Finally there is **Windows** and **Help** used to close wind getting help and debug respectively.

3.1.2 Location and navigation bar

The location bar is the place or location which you are currently viewing. It has standard navigation buttons like *Back*, *Forward*, *Up* and *Refresh*.







Figure 3.2: Location bar

You can drop icons and other URLs (or URI compatible strings) into this location. For example you can drag windows icons (from your windows desktop) and internet browser links (from for example Internet Explorer or Firefox) into this location bar. Also you can start a drag from the mini icon in the location bar and drop resources into the ResourceTree, IconsPanel or on your native desktop. The action of a VBrowser to native desktop Drag and Drop (DnD) might depend how your local operating settings are configured.

You can also drag files directly from your local desktop into the ResourceTree and IconsPanel windows.

Navigation Buttons

Here is an overview of the navigation buttons:

-  Refreshes the current viewed location.
-  Browses back one location from history.
-  Browses a location forward again, when the user has browsed back.
-  Browse to the parent location of the current viewed location.






3.1.3 Toolbars

Toolbars are the bars which have a set of buttons with small icons in them. These buttons represent menu shortcuts or optional ways to view the contents of a location.



Figure 3.3: Toolbars in the VBrowser

An overview of the current toolbar buttons is as follows:

-  Grid Credentials button. Status is invalid: Grid proxy has not been created.
-  Grid Credentials button. Status is valid: Grid proxy has been created.
-  Icons panel button.
-  List panel button.
-  Table panel button.

Pressing the icon button will show the resources as icons, pressing the table icons will present the resources in table with detailed information about the resources.

3.1.4 Table panel

When clicking on the table panel icon details from the resources will be shown in a table form as can be seen in Figure 3.4.

For each resource the default attributes are shown. Clicking on the header of the table column will sort the table according to the logical value of the fields in that column.

More attributes can be added by right-clicking (or pressing alt-mouse button) on the headerbar of the table panel as can be seen in Figure 3.5.

You can also move around the columns by dragging the column header to the desired position. Currently the layout of the table is not saved between browsing sessions.

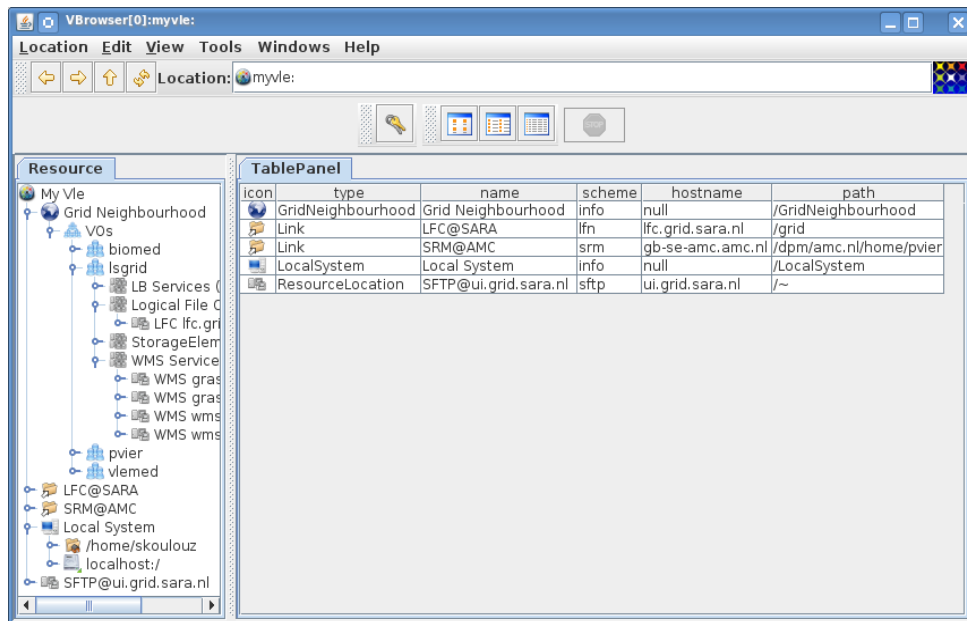


Figure 3.4: Table Panel

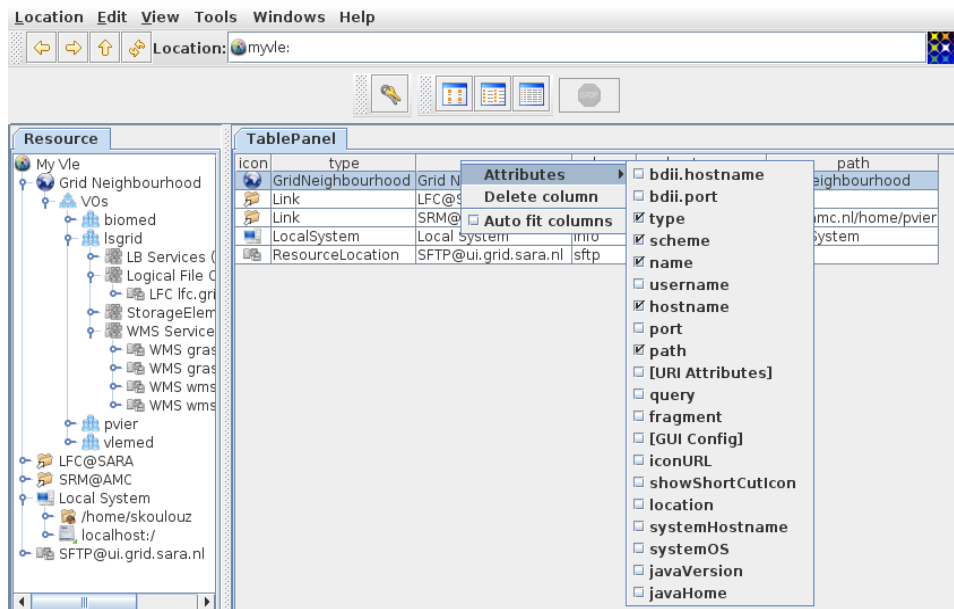


Figure 3.5: Table Panel with Attributes menu

3.1.5 Pop-up or Action menu

When right-clicking (or use alternative mouse button) on a resource, an pop-up menu will appear. This is the Resource's Action Menu and from this menu special actions can be selected which are to be performed on the resource. These actions include standard Copy and Paste actions as well as more complicated actions. See Figure 3.6

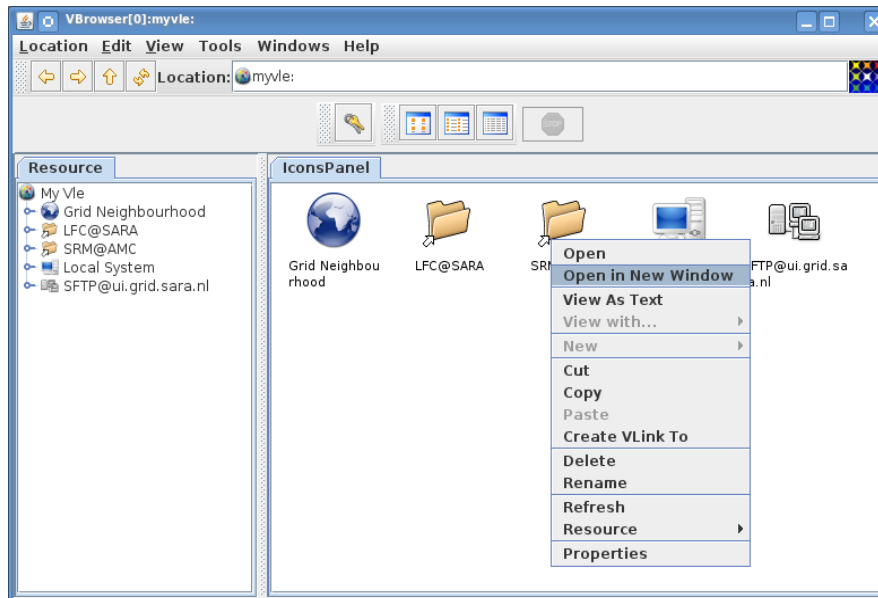


Figure 3.6: VBrowser Action Menu

3.2 Authenticating yourself with the Grid

The first thing you do is to authenticate yourself with the grid. This is done by creating a *grid proxy certificate* which is a file containing a temporary key which you must use to access resources on the grid. For security reasons this file can only be used for a limited time.

To create a grid proxy you can use your *Grid Certificate* and your secret *Passphrase* which combined can create a Grid Proxy for you.

To create one interactively, click on the *keys* icon which can be seen in Figure 3.7.



Figure 3.7: Grid Proxy Icon

A dialogue will appear where you can create your grid proxy. See Figure: 3.8.

To create a Grid Proxy, enter your passphrase and press **Create**. Your grid proxy will be valid for 12 hours if not specified otherwise. Also to be able to use resources assigned to your VO, you would have to enable the VOMS proxy¹ option and fill in your VO name. More information about VOs may be found here: [7, 2] You can enter grid proxy creation options, like proxy location (file path) and lifetime (in hours), in the dialog if needed. Also you can **Destroy** your grid proxy if you don't need it any more. Pressing **Create** again will refresh your proxy and update it with a new lifetime.

¹VO: Virtual Organization. VOMS: Virtual Organization Management Service

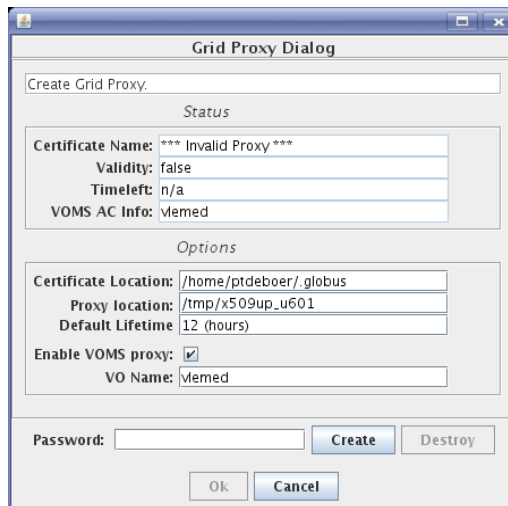


Figure 3.8: Grid Proxy Creation Dialogue

3.3 Grid Neighbourhood

The **Grid Neighbourhood** is a collection of Grid Resources (File Catalogue, Storage Elements, etc.) grouped by VO. It can be configured using the property panel specifying the host and port of the BDII service to query [5, 1].

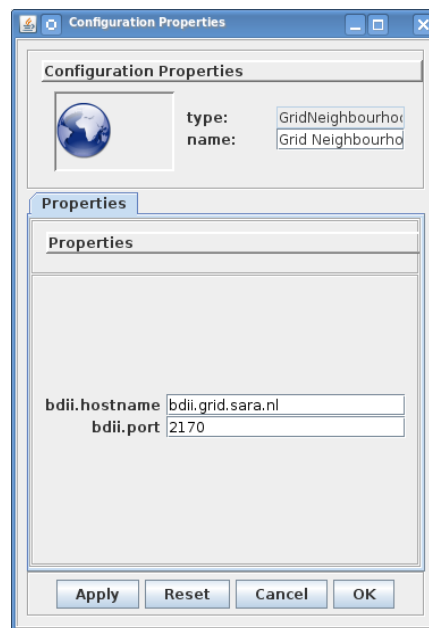


Figure 3.9: Grid Neighbourhood properties

This resource seen in Figure 3.10 is organized as following:

- **Grid Neighbourhood** . The top level resource.
- **VO Group** . The collection of VOs the user is part of. Each time the user authenticates himself having the VOMS proxy enabled, a new **VO Group** is created. Alternatively the user can create a new **VO Group** manually, by pressing **right-click** , **New, VO**. Moreover the current active VO is indicated with a different icon (see Figure 3.11).

- VO . The collection of resources that this VO has at its disposal.

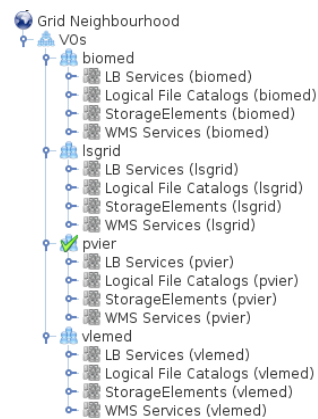


Figure 3.10: Grid Neighbourhood Resource hierarchy



Figure 3.11: The icon for the selected VO

Section 3.4 describes how to add more resources, that might not be published by the BDII service as well as how to customise resource settings.

3.4 Configuring your personal environment

When the **VBrowser** starts, the left panel will show the Resource Tree with as first entry the 'Root Resource' currently named: **MyVLe**. This root resource will contain your personal grid environment. You can rename this 'Toplevel Resource' to your any name you want.

3.4.1 Adding Resources

To add resource to your root resource, do the following:

- **right-click** on **MyVLe** and select:**New**
- Select the resource you want to add.

Deleting can be done as follows:

- **right-click** on the resource icon you want to delete and select:**Delete**

Note: Deleting a resource from the root resource (**MyVLe**) will delete the entry, never the directory it points to.

To configure the settings for a resource, select the properties from the pop-up menu as follows:

- **right-click** on the resource icon and select:**properties**

Check and optionally change the properties of this resource.

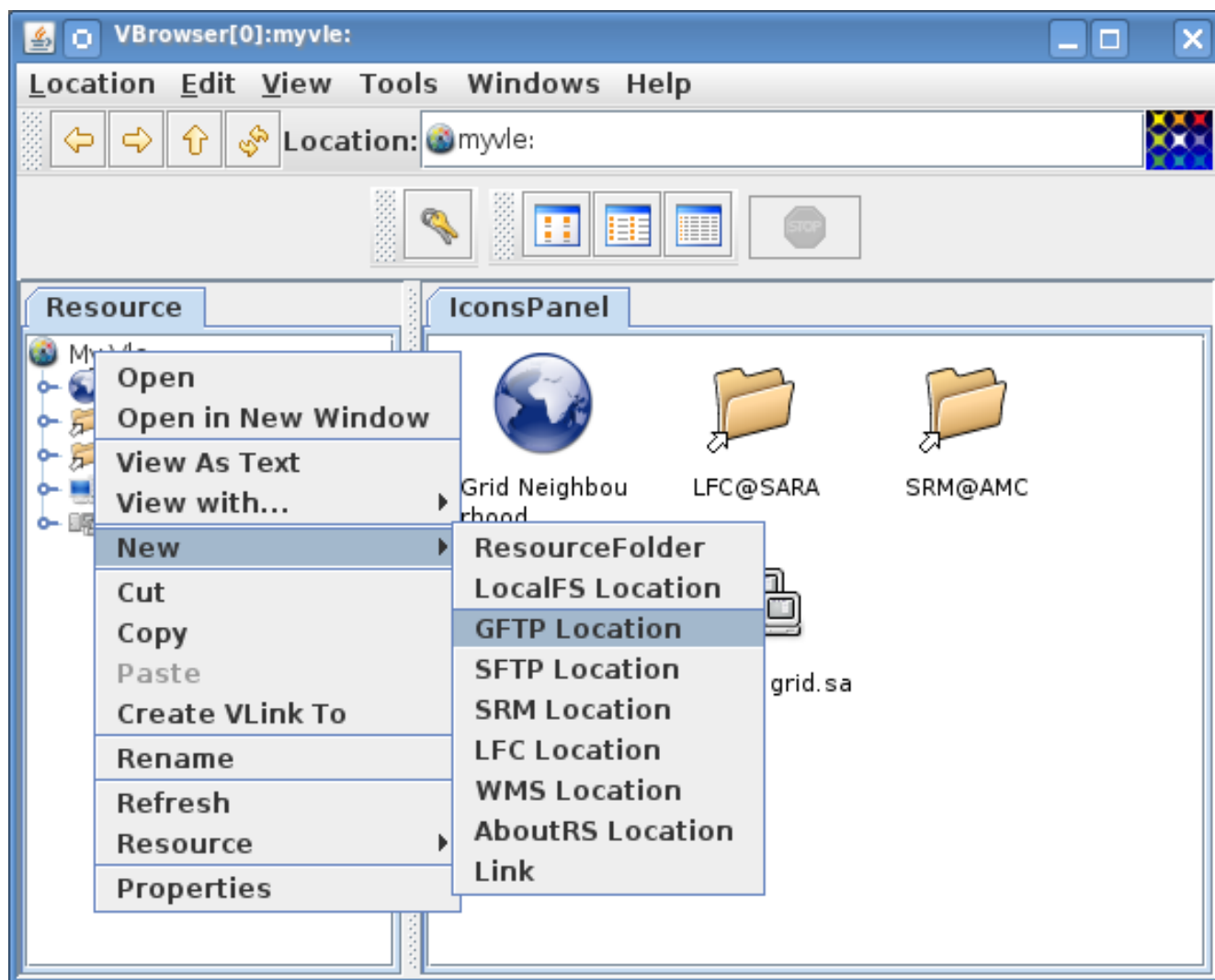


Figure 3.12: Create new location

3.4.2 Adding New (Grid) Resource Location

You can add a new Grid Resource Location, for example a 'GridFTP' location as follows:

- **right-click** on MyVLe and select:New →GFTP Location

Now set the GridFTP Location by selecting the properties option from the pop-up menu as follows:

- **right-click** on the GridFTP icon and select:properties

Resource Location Properties

type: ResourceLocation
name: New GFTP Location

Location | Server Settings | Icon Properties

Location Properties

Location:
scheme: gsiftp
hostname: gridftp.sara.nl
port: 2811
path: /~

URI Attributes: ☐ enable URI attributes
URI Query:
URI Fragment:

Apply Reset Cancel OK

Figure 3.13: GridFTP Location

Fill in the location of the new GridFTP server.

- *scheme* : the scheme to use. For GridFTP leave this to 'gsiftp'.
- *host* : the hostname of the GridFTP location.
- *port* : the port of the GridFTP location. The default value is '2811'
- *path* : The path to start browsing. Leave to '~' to use your default home location on the remote server.

- *URI properties* : Not supported for GridFTP.

To configure the server properties for this location, click on the **Server Settings** tab and specify the settings for this server.

Resource Location Properties

type: ResourceLocation
name: New GFTP Location

Location **Server Settings** **Icon Properties**

Server Settings

server URI: gsiftp://gridftp.sara.nl:2811/

passiveMode: true
allow3rdParty: true
AUTH_SCHEME: GSI_AUTH

New Delete

Apply Reset Cancel OK

Figure 3.14: GridFTP Server Settings

Current Server Settings for GridFTP servers are:

- *passiveMode* : set to 'true' when incoming connections are not possible, for example when browsing from behind a firewall.
- *allow3rdParty* : whether 3rd party copying is possible to and from this server. This is used when copying between two (remote) Grid FTP locations.

3.4.3 Adding an LFC Location

To create an LFC [3] location do as follows:

- **right-click** on MyVLe and select: New → LFC Location

After creating a new location the resource properties editor should pop-up. You can also manually edit these properties by selecting the edit properties option from the pop-up menu as follows:

- **right-click** on the LFC icon and select:properties

The screenshot shows a 'Resource Location Properties' dialog box. At the top, there's a title bar and a sub-title 'Location Properties'. Below the title bar, there's a small icon of a server and a 'type' field set to 'ResourceLocation'. The 'name' field is 'New LFC Location'. Below this, there are three tabs: 'Location', 'Server Settings', and 'Icon Properties'. The 'Location' tab is selected. Inside the 'Location' tab, there are fields for 'scheme' (lfn), 'hostname' (lfc.grid.sara.nl), 'port' (5010), and 'path' (/~). There's also a checkbox for 'enable URI attributes' which is unchecked. Below this are fields for 'URI Query' and 'URI Fragment'. At the bottom of the dialog are four buttons: 'Apply', 'Reset', 'Cancel', and 'OK'.

Figure 3.15: LFC Location


Fill in the location of the new LFC server.

- *scheme* : the scheme to use. For LFC leave this to 'lfn'.
- *host* : the hostname of the LFC server.
- *port* : the port of the LFC server. The default value is '5010'
- *path* : the path to start browsing. Fore example '/grid/lsguid' or use tilde: '/ ~' to auto resolve to you VO home location.
- *URI properties* : Not supported for LFC locations.

To specify the server configuration for this location, click on the **Server Settings** tab and specify the settings for this server. Here an overview of the LFC settings:

- *listPreferredSEs* : list of Preferred Storage Elements to use when selecting a replica to read from or when creating a new replica. This is a comma seperated list of hostnames which can have an optional port number.

Resource Location Properties



type: ResourceLocation

name: New LFC Location

Location

Server Settings

Icon Properties

Server Settings

server URI: lfn://lfc.grid.sara.nl:5010/

listPreferredSEs srm.grid.sara.nl,tbn18.nikhef.nl

replicaSelectionMode Preferred

replicaCreationMode Preferred

replicaNrOfTries 5

subDirDateScheme yyyy-MM-dd

generatedDirname metgenerated

replicaNamePolicy Similar

AUTH_SCHEME GSI_AUTH

New

Delete

Apply

Reset

Cancel

OK

Figure 3.16: LFC Server Settings

- *replicaSelectionMode* : replica selection algorithm used when there is more than one replica to choose from (used when reading an LFC file).
- *replicaCreationMode* : Storage element selection algorithm used when creating new replicas (used when uploading or writing to an LFC file).
- *replicaNrOfTries* : number of attempts tried when selecting a replica to read from or when creating a new replica at a Storage Element. A minimum of 3 is recommended or the number of replicas+1 if that is bigger than 3.
- *generatedDirname* : parent directory which will be used at a Storage Element for all generated replicas. This directory name will be appended to the default VO writeable location or VO Storage Area (see explanation below).
- *subDirDateScheme* : sub directory syntax used when replicas are created in the replica parent directory: *generatedDirname* (see explanation below).
- *replicaNamePolicy* : file name policy when creating replicas at Storage Elements (see explanation below).

Each of the above server properties is explained more in details below:

List of Preferred Storage Elements (*listPreferredSEs*)

The server property *listPreferredSEs* is a comma separated list of hostnames which contains the user preferred Storage Elements. This list is used when uploading a file to an LFC server to choose a Storage Element to store a replica. When downloading a file, this list is used to select a replica which matches a Storage Element from this list. How this matching occurs depends on the *replicaSelectionMode* or the *replicaCreationMode* settings.

Replica Selection Algorithm (*replicaSelectionMode*)

The option *replicaSelectionMode* determines which algorithm is used when selecting a replica to read from. This is one of the following:

- *Preferred*¹ : Try to find a replica which matches one of the hostnames listed in the server property which specified the list of preferred Storage Elements: *listPreferredSEs*. The order in which replicas are tried is the same as the order of the hostnames in this list.
- *PreferredRandom*² : Same as *Preferred* but the list of matching replicas is randomized.
- *AllSequential* : Try all replicas in order as they are registered at the LFC server. The first that is available will be used. If all replicas have been tried, the selection procedure will start with the first until the maximum number of tries has been reached.
- *AllRandom* : Same as *AllSequential* but the order in which replicas are tried is randomized. As this randomization is 'true' random, same replicas might be tried twice in a row. This is by design.

¹ If no replicas match the list of preferred Storage Elements (*listPreferredSEs*) or this list is empty, the mode *AllSequential* will be used.

Replica Creation Algorithm (`replicaCreationMode`)

The option *replicaCreationMode* determines which algorithm is used when selecting a Storage Element to create a replica. This is one of the following:

- *Preferred* : Only the Storage Elements from the *listPreferredSEs* list are used to create a new Replica. The order in which Storage Elements are tried is the same as the order of the list.
- *PreferredRandom* : Same as *Preferred* but the list will be randomized. Since the randomiser is 'true' random, a Storage Element might be tried multiple times even if it has already been tried before unless it already has a replica.
- *DefaultVORandom* : A Random Storage Element is used from the list of allowed Storage Elements for the current user's VO by querying the grid info service (BDII). If this Storage Element fails another is randomly selected.

Replica number of Tries (`replicaNrOfTries`)

The number of attempts undertaken to either read a replica or the number of attempts to create a replica at a Storage Element.

When reading from an LFC file (downloading), this number determines how many times an attempt is made to read from any of the available replicas. If, for example, the number of replicas is only 1, the same replica will be tried each time. If the number of available replicas is less than the *replicaNrOfTries* the same replica might be tried more than once. Which replica is used depends on the *replicaSelectionMode*.

When uploading a file to LFC, this number determines how many times an attempt is made to create a replica at one of the available Storage Elements. Which Storage Element is used depends on the *replicaCreationMode*.

Replica directory and file name creation policy (`generatedDirname`, `subDirDateScheme`, `replicaNamePolicy`)

The options *generatedDirname*, *subDirDateScheme* and *replicaNamePolicy* determine which path is used when creating a replica at a Storage Element.

The path is created by using the following syntax:

```
srm://<VO Storage Area>/<generatedDirname>/<subDirDateScheme>/<replicaFileName>
```

The `<VO Storage Area>` is the VO allowed Storage Area which is dynamically determined by querying the appropriate grid info service (BDII).

The `<replicaFileName>` is a generated file name which can resemble the logical file name of the LFC entry the file is an replica for. This depends on the *replicaNamePolicy* setting.

This can be one of the following:

- *Random* : The filename is a randomized file name only.
- *Similar* : The filename contains the original logical filename and a randomized identifier.

Example of a *Random* replica file path at a storage element:

```
/pvier/vletgenerated/2009-07-01/file_fad71348-d80b-48e5-bf2d-24590bb7c5f2
```

Example of a *Similar* replica file path for the LFC file with name 'experiment.txt' :

```
/pvier/vletgenerated/2009-07-01/experiment_txt_fad71348-d80b-48e5-bf2d-24590bb7c5f2
```

3.5 Links or shortcuts

Another way of creating a resource entry is creating a **Link**. A **Link** behaves in a similar way as shortcuts on windows. This type of resource does *not* resemble a Unix hard- or softlink in anyway. When opening a **Link**, the location pointed to by the **Link**, will be opened.

Typically a **Link** icon has a small arrow in the bottom left corner to indicate it is a link, as can be seen in Figure 3.17.



Figure 3.17: Link icon

You can create a link by selection the **Create VLink to** option from the pop-up menu or action menu.

Chapter 4

Other Tools

4.1 Application and Tools

Other applications currently available in the VL-e Toolkit are:

- **GridProxyDialog**: for creating and managing Grid Proxies.
- **VLTerm**: simple vt100 emulator with some xterm extensions.
- **uricopy.sh,urils.sh,uristat.sh**: URI copy, list and stat script.
- **jython.sh**: Jython startup script.

4.2 GUI utils

4.2.1 GridProxyDialog

Some part of the **VBrowser** can be used stand alone. The Grid Proxy Init dialog can be called by running the **GridProxyDialog.jar** as follows:

```
java -jar $VLET_INSTALL/bin/GridProxyDialog.jar
```

Or double click on the jar file in the **VLET_INSTALL/bin** directory:

```
VLET_INSTALL/bin/GridProxyDialog.jar
```

You cannot move this file out of the installation, but you can create a shortcut to the jar file instead.

4.2.2 VLTerm

The VL-e Toolkit has an beta version of a VT100 terminal emulation program. This terminal emulator can be used as a backup if there is no standard xterm or SSH terminal program available on the user's

Most VT100 and some XTERM control codes are supported. For basic remote command execution this terminal application can be used.

```
java -jar $VLET_INSTALL/bin/vlterm.jar
```

VLET_INSTALL/bin/vlterm.jar

```

X VLTerm: <2>
Session Settings Help
drwxrwxr-x 2 ptdeboer ptdeboer 4096 Jul 24 16:26 certificates
drwxrwxr-x 2 ptdeboer ptdeboer 4096 Jul 24 16:26 fmri
drwxrwxr-x 10 ptdeboer ptdeboer 4096 Jul 24 16:26 old
drwxrwxr-x 2 ptdeboer ptdeboer 4096 Jul 24 16:26 testData
-rwxr-xr-x 1 ptdeboer ptdeboer 0 Jul 24 16:27 vbrowser
lrwxrwxrwx 1 ptdeboer ptdeboer 34 Jul 24 16:27 vbrowser.sh -> /opt/vl-e
/vlet_0.6/bin/vbrowser.sh
[ptdeboer@ui ptdeboer]$ ls
certificates fmri old testData vbrowser vbrowser.sh
[ptdeboer@ui ptdeboer]$
[ptdeboer@ui ptdeboer]$ ls
certificates fmri old testData vbrowser vbrowser.sh
[ptdeboer@ui ptdeboer]$ ll
total 16
drwxrwxr-x 2 ptdeboer ptdeboer 4096 Jul 24 16:26 certificates
drwxrwxr-x 2 ptdeboer ptdeboer 4096 Jul 24 16:26 fmri
drwxrwxr-x 10 ptdeboer ptdeboer 4096 Jul 24 16:26 old
drwxrwxr-x 2 ptdeboer ptdeboer 4096 Jul 24 16:26 testData
-rwxr-xr-x 1 ptdeboer ptdeboer 0 Jul 24 16:27 vbrowser
lrwxrwxrwx 1 ptdeboer ptdeboer 34 Jul 24 16:27 vbrowser.sh -> /opt/vl-e
/vlet_0.6/bin/vbrowser.sh
[ptdeboer@ui ptdeboer]$ ls
certificates fmri old testData vbrowser vbrowser.sh
[ptdeboer@ui ptdeboer]$

```

30

4.3 Command line tools

4.3.1 URI script interface

At this moment there are three scripts which can interface with the Virtual Resource System (VRS). These scripts are installed in **VLET_INSTALL/bin/** and are:

- **uricopy.sh** : copy remote file and directories.
- **urils.sh** : list remote resources. Also non filesystems can be listed.
- **uristat.sh** : like the unix 'stat' command this script can be used to provide status information about the resource. Any URI can be used.

uricopy.sh

The **uricopy.sh** syntax is follows:

```
uricopy.sh [options] <source URI> <destination URI> [-D<PROPERTY>=<VALUE >]*
```

Command line arguments are:

- **<source URI>** : source file or directory.
- **<dest URI>** : parent directory to copy to. The source file or directory is always created as child of this parent directory.
- **[options]** can be:
 - proxy <proxyfile> ; Optional proxy file
 - r | -dir ; perform a recursive copy of the source location (for copying directories).
 - v [-v] ; be verbose. provide option twice to be even more verbose.
 - f | -force ; overwrite (optional) existing target location.
 - move ; move resource and delete source URI after copy command.
 - mkdirs ; create target directory path
 - result ; print resulting destination URI as follows: "result=...".
 - debug ; enable debug output.

The properties which can be specified are depending on the source and destination URIs. Most VLET settings can be specified as command line options using the **-D<NAME>=<VALUE>** syntax.

An overview of relevant properties are:

Global properties:

-DpassiveMode=true always use passive mode for all file transfers.

General usage:

The source and destination URIs are mandatory. Options can be both specified before and after the URIs. The source URI must be an existing file or directory and the target URI *must* be an existing directory! This is to avoid ambiguity between destination files and directories. The new file or directory will always be created as a child entry in the target directory.

Copying directories:

To copy a directory specify the `-r` option (for recursive copy).

The uricopy command will create the new destination directory which always will be a child (subdirectory) of the destination URI.

Moving files or directories:

To move a file or a directory specify the `-move` option. The source will be deleted after and *only* after a successful copy.

urils.sh

```
urils.sh [options] [ -proxy <proxyFile> ] <URI>
```

Command line argument are:

- `<URI>` : remote resource. Can be any 'listable' object.
- `[options]` can be:
 - `-proxy <proxyfile>` ; Optional proxy file
 - `-l` ; long list like 'ls -l'.
 - `-guid` ; print GUID if the resource has it.
 - `-noresolve` ; do not resolve links.
 - `-vrls` ; print full VRLs instead of resourcenames.

uristat.sh

```
uristat.sh [options] [ -proxy <proxyFile> ] <URI>
```

Command line argument are:

- `<URI>` : remote resource. Can be anything.
- `[options]` can be:
 - `-proxy <proxyfile>` ; Optional proxy file
 - `-v [-v]` ; be verbose (use twice for more)
 - `-file` ; print out common file attributes
 - `-all` ; print out all resource attributes (default)
 - `-[no]quotes` ; put value between single 'quotes' (or not)
 - `-checksum <TYPE>` ; report checksum (if supported)
 - `-attrs=<attributelist>` ; list of comma separated VRS Attribute names
 - `-D<prop>=<value>` ; specify java properties to the JVM

4.3.2 Examples

LFC uricopy example

Below an example how to use uricopy to upload a file to an LFC server.


```
$VLET_INSTALL/bin/uricopy.sh file:/home/ptdeboer/hello.txt \
lfn://lfc.grid.sara.nl/grid/pvier/piter
```

The backspace in the above example means the next line should be concatenated, without an newline, the the previous.

Options or server settings can be provided as either a Java property or a URI attribute. For example to specify the preferred Storage Elements provide the *lfc.listPreferredSEs* option to the uricopy script:

```
$VLET_INSTALL/bin/uricopy.sh file:/home/ptdeboer/hello.txt \
lfn://lfc.grid.sara.nl/grid/pvier/piter\
-Dlfc.listPreferredSEs=srm.grid.sara.nl,tbn18.nikhef.nl
```

The above command will try to register the file at the first Storage Element in the list, if that fail it will try the second storage element.

Optional LFC properties can be:

```
-Dlfc.listPreferredSEs=SEHost1,SEHost2 ; list of preferred Storage Elements
-Dlfc.replicaNrOfTries=5                ; number of retries when reading/creating
                                         a replica.
-Dlfc.replicaCreationMode=Preferred     ; one of: Preferred, PreferredRandom,
                                         DefaultVO, DefaultVORandom
-Dlfc.replicaSelectionMode=Preferred   ; one of: Preferred,PreferredRandom,
                                         AllSequential, AllRandom
```

SRM file listing example

If a directory, for example a replica directory at a remote (SRM) storage element, contains a lot of files, it is possible to list the directory in parts using an 'offset' and a 'count' parameter.

Use `srmCount` and `srmOffset` parameters to specify the maximum number of files you want to list and the offset.

For example:

```
$VLET_INSTALL/bin/urils.sh srm://srm.grid.sara.nl:8443/data/pvier/ptdeboer/bigdir?\
srmCount=100&srmOffset=200
```

4.3.3 Customized uricopy script

You can use the `uricopy.sh` as an example and modify it to your liking. Make sure the `VLET_INSTALL` environment variable is set so the script can find the installation as follows:

```

#!/bin/bash
##
# File   : Example modified uricopy.sh script
# Version: VLET 1.5
# ---

if [ "$1" == "" ] ; then
    echo "Please specify VRLs"
    exit 1
fi

# Default installation:
export VLET_INSTALL=/opt/vlet
# Configuration files on PoC R3:
#export VLET_SYSCONFDIR=/etc/vlet

# set base directory
BASE_DIR=$VLET_INSTALL

# my options:
OPTIONS="-DpassiveMode=true -r -force -info"

# source vlet settings:
source $VLET_INSTALL/etc/vletenv.sh

##
# VLET java class to start: VFSCopy
CLASS=nl.uva.vlet.vfs.VFSCopy

# JVM options:
JVMOPTS="-Dvlet.install.sysconfdir=$VLET_SYSCONFDIR -jar $VLET_INSTALL/bin/bootstrapper.jar"

# Start bootstrapper which does the rest
echo "Command:" $JAVA $JVMOPTS $CLASS $OPTIONS $@
$JAVA $JVMOPTS $CLASS $OPTIONS $@
# keep return value:
RETVAL=$?

#return exit code from VFSCopy
exit $RETVAL;

```

4.4 Jython

4.4.1 Jython start script: jython.sh

Since Jython is a pure java implementation of Python, Jython can be used to access to full VLET API. To run a Jython script, call the Jython bootstrap script from the VLET installation. The location of this bootstrap script is: **VLET_INSTALL/bin/jython.sh** and the syntax is as follows:

```
jython.sh [-i] <jython file>
```

Where option **-i** starts the Jython interpreter in interactive mode (leave out the file when starting the Jython interpreter in this mode).

Jython examples can be seen in: **VLET_INSTALL**/py.

The documentation for the VLET API can be found at: **VLET_INSTALL**/doc/api/index-all.html.

Below an example how to use the VFS interface from VLET in a Jython script:

```

#!/opt/vlet/bin/jython.sh
##
# VFSCClient jython interface example
# (C) www.vl-e.nl
##
# File      : vfsclient.py
# Vlet version : 1.5
# Author    : Piter T. de Boer
#
# Info :
#   VLET vfs jython interface example.
#   To start execute this jython file, use:
#       $VLET_INSTALL/bin/jython.sh <jython.file>
##
import sys
#import VRS objects + VFS Client:
from nl.uva.vlet.vrl import VRL
from nl.uva.vlet.vfs import VFile, VDir, VFSCClient

### Helper methods
def boolstr(value):
    if value == True:
        return "true"
    else:
        return "false"

# creat new VFSCClient
vfs=VFSCClient()

# get Virtual File object:
dir=vfs.getDir("lfn://lfc.grid.sara.nl:5010/grid/pvier/");
print "Remote LFC directory = "+dir.toString()
print "--- Directory Attributes ---"
print " Directory modification time =",dir.getModificationTime()
print " Directory access permissions = \""+dir.getPermissionsString()+"\"

#get contents of remote Directory
contents=dir.list()

# array acces:
print "First file = ",contents [0];

# define print function:
def PrintNode(prefix,node):
    print prefix+"["+node.getType()+"] "+node.toString();

# apply MyPrint on contents array
print "--- contents of remote directory ---"
[PrintNode(" - ", file) for file in contents]
print "--- ---"

# get local home
home=vfs.getDir("file:///~");
print "Local home=",home;

```

```

### remote navigation
# get remote file:
fileVRL="lfn://lfc.grid.sara.nl:5010/grid/pvier/piter/test.txt";
vrLObject=VRL(fileVRL)

# dissect VRL (=URI)
print "-- VRL object ---"
print " VRL String      =",fileVRL;
print " VRL Object      =",vrLObject;
print " VRL Hostname     =",vrLObject.getHostname();
print " VRL Port         =",vrLObject.getPort();
print " VRL Path          =",vrLObject.getPath();
print " VRL Extension    =",vrLObject.getExtension();
print "---"

if (vfs.existsFile(fileVRL) is True):
    print "File exists:"+fileVRL;
else:
    print "*** Error: Remote file does not exists:"+fileVRL;

# get/create Virtual File object of remote file
remoteFile=vfs.getFile(fileVRL);
print "--- File Attributes ---"
print " modification time =",remoteFile.getModificationTime()
print " length              =",remoteFile.getLength()
print " access permissions = \"\""+remoteFile.getPermissionsString()+"\"\"
print " isReadable          =",boolstr(remoteFile.isReadable())
print " isWritable           =",boolstr(remoteFile.isWritable())
print " mimetype             =",remoteFile.getMimeType()

#get contents of remote Directory

# getContents() returns byte array, getContentsAsString returns String object
text=remoteFile.getContentsAsString();
print "Contents of remote file = "+text;

# Navigating example: "cd .."
remoteParent=remoteFile.getParent();
print "Remote parent = ", remoteParent;

# example "VDir.hasFile()" (hasDir/hasChild)
if (remoteParent.hasFile(remoteFile.getBasename()) is True):
    print "Remote parent reports that is has the remote file as child"
else:
    print "***Error: Parent directory of remote file reports is doesn't have the Child!"

# copy to local home directory, overwrite existing:
resultFile=remoteFile.copyTo(home);
print "Copied remote file to:",resultFile;
print "Check whether file is local:",resultFile.isLocal();
print "Local path of file is:",resultFile.getPath();

```

```
# get system temp dir:
print "tempdir'",vfs.getTempDir();

# create Unique Temp Dir() on local home:
tmpdir=vfs.createUniqueTempDir();
print "unique tempdir =",tmpdir;

print "end."
sys.exit(0);
```

See the examples directory `VLET_INSTALL/py/*` for more jython scripts.

Chapter 5

Customization

5.1 Custom viewers/plugins

The VBrowsers supports two kind of extension or plugins. One is for protocol implementations or 'VDrivers', The other are VBrowsers Viewer extensions or 'Viewer Plugins'. The first are mapped to URI schemes like 'srm://' the latter are mapped to resource mimetypes like 'text/plain'.

You can install extra viewers or VBrowsers plugins in the following directories (create them if they don't exist):

- **VLET_INSTALL/lib/plugins:** for installation plugins.
- **HOME/.vletrc/plugins:** path for user plugins.

A VBrowsers plugin can be a single jar file or a directory that contains the needed jar file and (optional) used libraries. The filename or directory must be the full classname of the plugin. This name triggers the VBrowsers upon startup to load the plugin. For example the custom VTK image viewer which can be install at one of the following locations:

```
VLET_INSTALL/lib/plugins/nl.uva.vle.app.vtk.viewer.NiiViewer/
```

for system wide installation of custom plugins, or:

```
HOME/.vletrc/plugins/nl.uva.vle.app.vtk.viewer.NiiViewer/
```

for custom plugins installed for a single user only (in his or hers HOME directory).

Some viewers might need extra configuration. For example in the case of the above VTK viewer, specifying the VTK library paths as follows:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/vl-e/vtk.5.0.2/lib:/opt/vl-e/mesa3d.6.4.2/lib
```

The above library paths are the installation paths as specified on the VL-e PoC environment (R2). You can add them to your `.bashrc` or add them to the installation configuration in `VLET_INSTALL/etc/vletenv.sh` if library paths are not already setup as specified.

Extra plugins can be downloaded from the sourceforge site at the 'files' section:

<http://sourceforge.net/projects/vlet/files/>

5.2 Custom Mimetypes and icons

Mimetypes are a way of telling an application what kind of content is in the file or resource. In most cases this is a simple mapping of an extension to a standard defined mimetype. This mimetype is a simple string which describes the type of content, for example “text/plain” or “image/jpeg”. Usually it consists of a generic part (“text/” or “image/”) and a specific part (“plain” or “jpeg”) separated by a (forward) slash.

The following files defines the mimetypes used in VLET:

- **VLET.INSTALL/etc/mime.types**: for installation configured mime types.
- **HOME/.vletrc/mime.types**: for user configured mime types.

The syntax of the mimetype configuration is the mimetype name followed by a list of extensions or complete filename(s) as follows:

type/subtype EXT1 EXT2 EXT3...

An example of some standard mimetypes is given as follows. First the mimetype name is given followed by their extension(s):

```
[170]
###
# File      : mime.types
# Location: $\VLETINSTALL/etc/mime.types
# ---
# Default mime types:  <MIMETYPE>  <EXTENSION1> <EXTENSION2> ...
#
image/x-xbitmap          xbm
image/x-xpixmap          xpm
image/jpeg               jpeg jpg jpe JPG
text/rtf                 rtf
text/plain               txt
# some custom mimetypes examples:
application/jglite-jobids vljids
application/taverna-scuffle scuffle
application/feat-fsf      fsf FSF

#end default mime types file
```

5.2.1 User defined mimetypes

A user can override the default mimetype for a resource or specify a new one by adding a line in the **HOME/.vletrc/mime.types** file (if the file is not there the user can created) as follows:


```
##
# File      : mime.types
# Location: $HOME/.vletrc/mime.types
# ---
# User defined mime types

# customized mime type for VL-e Text:
application/vle-text      vletxt txt TXT inf info

# end user defined mime types
```

Now .txt (and .TXT, .inf, .info) files will have the "application/vle-text" mimetype as can be seen in Figure:5.1

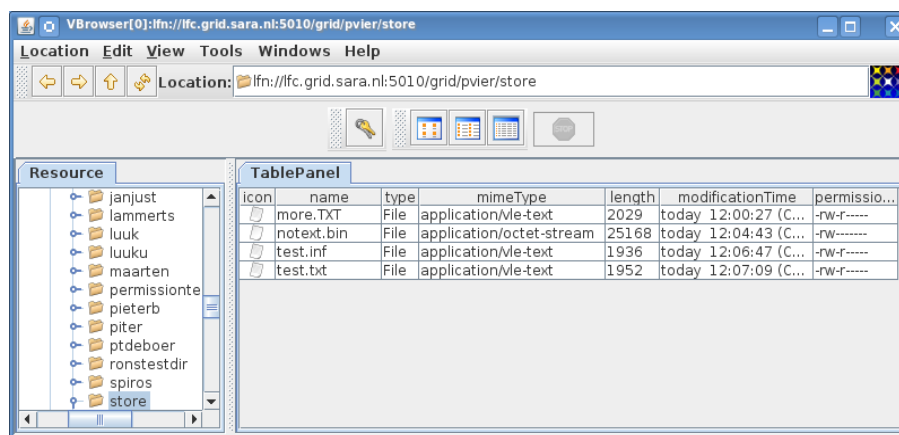


Figure 5.1: Custom Mimetypes

5.2.2 Web server mimetypes

When browsing websites and viewing resources at web servers, the web server has the responsibility to return the correct mimetype for the viewed resource. For example an html file has the mimetype "text/html" but the URL might not always point to a location which ends with '.html'. Also a query or server side script (php) might return content which differs from what might be expected when inspecting the URL, which for example might end in ".php". Configuring a web server to return custom mimetypes can be done in different ways and is highly dependant on the used web server and (Linux) environment it runs in.

Below configuration files for a Linux+Apache 2.0 setup:

- /etc/mime.types Global system (Linux) mime.types file.
- /etc/httpd/conf/httpd.conf Custom HTTPD configuration file or:
- /etc/apache2/httpd.conf Custom apache 2.0 configuration file.

The `/etc/mime.types` follows the same syntax as described in the previous paragraph. Changing this file will change global settings for the whole (Linux) system.

Below an example of extra configuration settings which can be added to the `httpd.conf` file. The location of this file depends on used web server and Linux distribution. See the documentation of the used web server for the correct location of this file.

For example, add the following custom mimetypes to the `httpd.conf` file to allow (glite) job monitoring resources to have the correct mimetype:

```
###
# Add custom VBrowser mime types.
# Add glite jobids:
AddType application/glite-jobids .vljids
# Add job output files ".out" and ".err" as plain text:
AddType text/plain .out
AddType text/plain .err
```

After changing this file the web server needs to be restarted for the changes to have effect.

5.2.3 Magic types

Another way to determine the file content is by matching the first bytes of a resource with well known 'magic types' which specify the type of file. For example Linux executables have the letters 'ELF' in the beginning of the file which specifies it is an Linux Executable.

Other examples are: GIF files (.gif) start with the ASCII letters 'GIF' and WAV files (.wav) start with the letters 'RIFF' and have the letter 'WAVE' in the RIFF header.

Magic types are not supported (yet) by the VBrowser but are available in the VLET Java API through the MimeTypes class (`nl.uva.vlet.util.MimeTypes`) and can be used by, for example, plugins to further determine the (actual) file contents.

5.2.4 Custom icons

Each resource which has a mimetype is mapped to a (default) icon based on the mimetype. The name of the icon is the mimetype with forbidden characters (like a forward slash) substituted with a dash ("-") and the extension ".png" is added. For example the mimetype: "application/vle-text" is transformed to the icon name: "application-vle-text.png"

If this icon exists in either the installation directory for mimetype icons or the user directory for mimetype icons, this icon will be used. To add mimetype icons, add the icon to one of the following directories:

`HOME/.vletrc/icons/mimetypes`

Use the above location for user configured icons. To make the icons available for all users, use the installation directory below:

`VLET_INSTALL/lib/icons/mimetypes`

In the example of the custom mimetype "application/vle-text" you can create a customized icon named: "application-vle-text.png" which should have the following path:

HOME/.vletrc/icons/mimetypes/application-vle-text.png

All text files will now have the customized 'vle' icon as can be seen in Figure 5.2

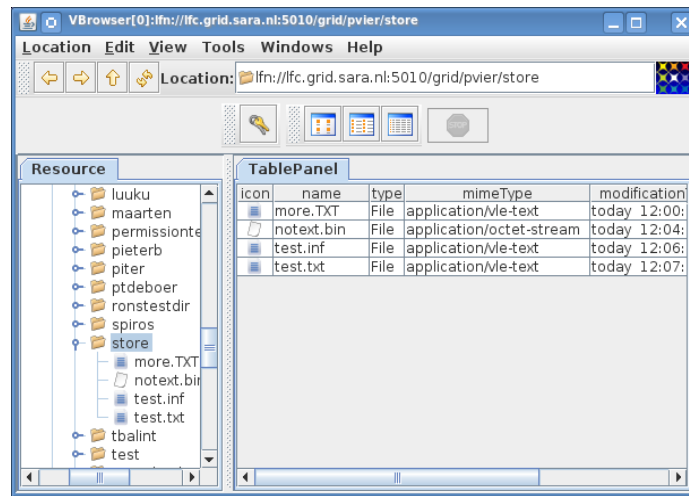


Figure 5.2: Custom Icons

You can also put the icon in your installation by creating it at the following path:

VLET_INSTALL/lib/icons/mimetypes/application-vle-text.png

5.2.5 Viewer preferences and defaults

To specify which viewer (vbrowser plugin) is used the user can configure the file HOME/.vletrc/viewerconf.prop. This file specifies per line the mimetype and the viewerclass which is started as default viewer when the user opens or clicks (on) a resource. The syntax is as follows:

mimetype/mimesubtype=viewerclass

For example to specify a new viewer for the “application/vle-text” mimetype, add a line as follows:

```
##
# File      : viewerconf.prop
# Location: $HOME/.vletrc/viewerconf.prop
#---
# Mimetype/viewer class mapping file
# <MIMETYPE>=<VIEWERCLASS>

# viewerclass mapping for the application/vle-text mimetype:
application/vle-text=nl.uva.vlet.gui.viewers.HexViewer

# end viewerconf.prop file
```

Now, instead of the default textviewer, the actual bytes will be shown in the HexViewer utility.

Other default vbrowser internal viewer classes are:

```
nl.uva.vlet.gui.viewers.TextViewer  
nl.uva.vlet.gui.viewers.HexViewer  
nl.uva.vlet.gui.viewers.VHTMLViewer  
nl.uva.vlet.gui.viewers.ImageViewer
```

For information about how to create custom plugins, see the VLET Developers Guide (ref:[]).

Appendix A

Appendices

A.1 VRL specification

This appendix specifies the syntax used for VRLs (Virtual Resource Locators) which follows the specification of URIs (Universal Resource Indicators) as specified in [4]. In short a VRL is an URI, but not all URIs are VRLs.

VRLs can be seen as Virtual URLs which may have extra schemes which do not map to an actual network protocol but to an 'Virtual' Protocol or 'VRS' Driver protocol (VDriver). An example is 'infor' and 'about' which are not an actual network protocols but trigger internal Resource Description Services.

A.1.1 Syntax

```
VRL                                ::= SCHEME + ':' + SCHEMESPECIFICPART

SCHEMESPECIFICPART ::=      [ '/' + AUTHORITY ]
                           + [ '/' + PATH ]
                           + [ '?' + QUERY ]
                           + [ '#' + FRAGMENT ]

AUTHORITY                       ::= [ userinfo + '@' ] + [ hostname + [ ':' + PORT ] ]

SCHEME                          ::= 'file', 'gftp', 'sftp', 'http', 'lfn', 'srm', ..

userinfo                         ::= USERNAME + [ '.' + DOMAINNAME ] + [ ':' + PASSWORD ]

USERNAME                        ::= UNRESERVED <see rfc3986>

DOMAINNAME                      ::= UNRESERVED <see rfc3986>

hostname                        ::= UNRESERVED <see rfc3986>

PASSWORD                        ::= UNRESERVED <see rfc3986>

QUERY                           ::= UNRESERVED <see rfc3986>

FRAGMENT                        ::= UNRESERVED <see rfc3986>

UNRESERVED                      ::= [ ALPHA | NUM | '-' | '.' | '_' | '~' ] *
```

Notes:

- It is not recommended putting plain text passwords in URIs as this is a security hazard.
- The AUTHORITY part start with two slashes '/' but is optional. When the scheme specific part starts with only one slash it has no authority part.

A.1.2 Examples of URIs

This section explains how URIs (VRLs) are used in VLET/VBrowser .
Below an example of a full URI.

```
foo://john_doe@example.com:8042/over/there?name=ferret#nose
\_/  \-----/ \-----/ \-----/ \_/
|      |           |           |           |
scheme authority   path       query    fragment
```

After the first colon (':') the *scheme specific* part starts. This part can contain an authority part which always starts with two slashes ('//').

For example:

```
sftp://skoulouz@ui.grid.sara.nl:22/data/home/skoulouz
```

The *authority* part can be omitted if for the scheme no authority is needed. In that case the scheme specific part may NOT start with two slashes, but only one in for example in the following URI:

(1) `file:/home/dexter/e-laboratory`

This is equivalent with an empty *authority* URI as follows:

(2) `file:///home/dexter/e-laboratory`

The above URI (2) will be normalized to (1) to indicate an empty authority part and remove superfluous slashes.

A.1.3 URI Attributes

An URI can have URI attributes specified in the *query* part. Some scheme implementations support URI attributes to specify extra options or parameters to the underlying implementation.

The syntax is as follows:

```
QUERY          ::= [ ParameterList ]

ParameterList ::= Parameter + [ '&' + ParameterList ]

Parameter      ::= Name + '=' + Value
```

Example of two URI Attributes in an LFC URI:

```
lfn://lfc.grid.sara.nl/grid/vlmed/stuff\
?lfc.listPreferredSEs=srm.grid.sara.nl\
&lfc.replicaSelectionMode=Preferred
```

(Remove the backslash '\' when concatenating the URI parts).

You can use this feature when invoking the `uricopy.sh` script or accessing the VRS programmatically.

Most properties which can be specified as Resource Properties using the VBrowser can be specified here although not all combinations have been tested and some are implementation specific. Check the documentation of the specific protocol implementation to see which property is supported.

Don't forget to prepend the property with the name of the scheme, like in '`lfc.listPreferredSEs`' to indicate the `lfc` property `listPreferredSEs`. That is when `lfc` is replicating a file, it will use the storage elements indicated after the '`lfc.listPreferredSEs=`' which for the example above would be `srm.grid.sara.nl`

A.2 Overview of configuration files and directories

The next sections show an overview of the distribution and user configuration files. See chapter 5 for the customization of these files.

A.2.1 Installation files and settings

Selection of configuration files and directories in the VLET distribution replace `VLET_INSTALL` with the location of the distribution):

<code>VLET_INSTALL/etc</code>	Directory where all distribution settings are.
<code>VLET_INSTALL/etc/vletrc.prop</code>	Installation properties and (default) settings.
<code>VLET_INSTALL/etc/certificates</code>	Extra trusted CA root certificates.
<code>VLET_INSTALL/etc/vomsdir/voms.xml</code>	VO Server configuration.
<code>VLET_INSTALL/lib/plugins</code>	Custom VDriver plugins and VBrowser Viewers.

A.2.2 Default installation configuration file `vletrc.prop`

The default settings for SRB and for LFC can be found in `VLET_INSTALL/etc/vletrc.prop`. These are the default properties used when creating a new resource or contact remote resources. Most of these properties can also be specified during startup or as extra URI attribute in the URI (VRL).

Default LFC settings

The default settings for LFC which can be found in `VLET_INSTALL/etc/vletrc.prop`.

```
## Default LFC Server settings:
##
# Default LFC hostname: Specify empty field for the user to fill in.
#lfc.hostname=lfc.grid.sara.nl
lfc.hostname=<LFCHOST>
# Default port:
lfc.port=5010
## Example of preconfigured Storage Elements:
#lfc.listPreferredSEs=srm.grid.sara.nl,tbn18.nikhef.nl
lfc.replicaNrOfTries=5
## Replica selection mode (reading).
## One of: Preferred, PreferredRandom, AllSequential, AllRandom
#lfc.replicaSelectionMode=PreferredRandom
## Replica creation mode (writing)
## One of: Preferred, PreferredRandom, DefaultVO, DefaultVORandom
#lfc.replicaCreationMode=Preferred
## replica naming policy. One of: Similar, Random
#lfc.replicaNamePolicy=Similar
```


A.2.3 User configuration files

All the user configurations are stored in the users home directory under `.vletrc`.

An overview of current used directories and files are (replace HOME with the user's home directory):

<code>HOME/.vletrc/</code>	Directory where user configuration files are stored.
<code>HOME/.vletrc/vletrc.prop</code>	User property settings.
<code>HOME/.vletrc/cacerts</code>	User trusted certificates.
<code>HOME/.vletrc/certificates/</code>	Directory for extra user trusted (CA) certificates.
<code>HOME/.vletrc/guisettings.prop</code>	User GUI (VBrowser) settings.
<code>HOME/.vletrc/mime.types</code>	User configured mime types.
<code>HOME/.vletrc/viewerconf.prop</code>	User mime type to Viewer mapping.
<code>HOME/.vletrc/myvle/</code>	User's 'My Grid' environment.
<code>HOME/.vletrc/icons/</code>	User customized icons.
<code>HOME/.vletrc/icons/mimetypes/</code>	User mimetype icons.
<code>HOME/.vletrc/icons/mimetypes/text-plain.png</code>	Example User customized icon for mime type 'text/plain'.
<code>HOME/.vletrc/plugins/</code>	User Installed plugins.

A.2.4 Example user customized mime.types file

The file `HOME/.vletrc/mime.types` contains user customized mime types.

See [5](#) how to customize this file.

```
[170]
###
# File      : mime.types
# Location: $HOME/.vletrc/mime.types
# ---
# Default mime types:  <MIMETYPE>  <EXTENSION1> <EXTENSION2> ...
#
# example vle text mime type:
application/vle-text          vletxt txt TXT
#
# example vlemmed mime types:
application/vlemmed-fslview    nii nii.gz NII NII.GZ
application/jglite-jobids      vljids
application/taverna-scuf1      scuf1
application/feat-fsf           fsf FSF
#end default mime types file
```

A.2.5 Example user customized viewerconf.prop file

The file `HOME/.vletrc/viewerconf.prop` contains user customized mime type to (VBrowser) Viewer mappings.

See [5](#) how to customize this file.

```
##
# File      : viewerconf.prop
# Location: $HOME/.vletrc/viewerconf.prop
#---
# Mimetype/viewer class mapping file. NO spaces between '=' !.
# <MIMETYPE>=<VIEWERCLASS>

# Example viewer mapping for the application/vle-text mimetype:
application/vle-text=nl.uva.vlet.gui.viewers.TextViewer

# Example vlemmed mimetype and to viewer mapping:
application/vlemmed-fslview=nl.amc.vlet.feat.fslview.gui.ViewerFSL
application/taverna-scufl=nl.uva.vlet.moteur.plugin.gui
application/jglite-jobids=nl.uva.vlet.monitoring.gui.ViewerGliteJobMonitoring
application/feat-fsf=nl.amc.vlet.feat.client.main.gui.ViewerFeatParameterSweep

# end viewerconf.prop file
```

Bibliography

- [1] Grid Information System. <https://svnweb.cern.ch/trac/gridinfo>, Aug 2010.
- [2] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, . Frohner, A. Gianoli, K. Lrentey, and F. Spataro. VOMS, an Authorization System for Virtual Organizations. In Francisco Fernandez Rivera, Marian Bubak, Andrs Gmez Tato, and Ramn Doallo, editors, *Grid Computing*, volume 2970 of *Lecture Notes in Computer Science*, pages 33–40. Springer Berlin / Heidelberg, 2004.
- [3] J.-P. Baud, J. Casey, S. Lemaitre, and C. Nicholson. Performance analysis of a file catalog for the lhc computing grid. *High-Performance Distributed Computing, International Symposium on*, 0:91–99, 2005.
- [4] T. Berners-Lee, R. Fielding, and L. Masinter. Rfc 3986, uniform resource identifier (uri): Generic syntax, 2005.
- [5] EGEE. Berkeley Database Information Index V5. <https://twiki.cern.ch/twiki/bin/view/EGEE/BDII>, Aug 2010.
- [6] Ian Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92, New York, NY, USA, 1998. ACM.
- [7] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.
- [8] RSA Laboratories. PKCS #1 RSA Cryptography Standard, June 2002.