CrossGrid Installation Guide

Roaming Access Server and Migrating Desktop

WP3.1

Document Filename:	**CG3.1_v3.0_CG_PSNC_install_guide_MD_RAS.doc**
Work package:	**WP 3.1- Portals and Roaming Access**
Partner(s):	**PSNC, DATAMAT**
Lead Partner:	**PSNC**
Config ID:	**CG3.1_v3.0_CG_PSNC_install_guide_MD_RAS**
Document classification:	**public**

Abstract:

This document is intended for the Site Administrator of CrossGrid testbed resources and may also be useful for people that need detailed information about the WP3 release.

Delivery Slip

	Name	Partner	Date	Signature
From	Mirosław Kupczyk	PSNC	11.2004	
Verified by				
Approved by				

Document Log

Version	Date	Summary of changes	Author
2-0	31/08/2004	Update RPM installation	Mirosław Kupczyk Paweł Wolniewicz
2-1 beta	15/09/2004	Update of JSS part	Marco Sottilaro
2-2	29/10/2004	Update after review.	Mirosław Kupczyk
3-0	10/11/2004	Update RAS part	Rafał Lichwała Mirosław Kupczyk Paweł Wolniewicz Marco Sottilaro
4-0	12/11/2004	Last corrections	Paweł Wolniewicz
4-1	20/12/2004	LCG Utils Configurations added	Marcin Płóciennik

CONTENTS

COPYRIGHT NOTICE

This research is partly funded by the European Commission IST-2001-32243 Project “CrossGrid”.

1. ABOUT THE SOFTWARE

1.1. SOFTWARE COMPONENTS

1.1.1. roaming access server installation

This document describes the installation procedure of the Roaming Access Server services (RAS). The Roaming Access Server is a part of the CrossGrid project architecture and provides a set of web services that are responsible for managing user and application profiles, job submission services etc.

The RAS software is delivered as a set of RPM binary packages, which can be installed using "rpm" tool available in RedHat distribution. After the successful RPM installation, some additional actions (described in this document) are required.

The Roaming Access Server (RAS) offers a well-defined set of web-services that can be used as an interface for accessing HPC systems and services (based on various technologies) in a common, standardized way. All communication bases on web services technology. This way we may support wide variety of client including personal computers, laptops, and in the future PDA, and mobile phones. It is one of the infrastructure features for supporting mobile users. The Roaming Access Server is a set of modules and plug-ins that provides interfaces to work with grids. It consists of several independent parts responsible for job submission, job monitoring, user profile management, data management, authorisation, and application information management.

1.1.2. MIGRATING DESKTOP installation

This document also describes requirements, configuration issues and installation procedure for the Migrating Desktop as an integrated tool which allows user friendly access to the Grid resources.

Migrating Desktop is an advanced user-friendly environment that serves as uniform grid working environment independent on specific grid infrastructure. Java based GUI is designed especially for mobile users and is independent on platform (MS Windows, Linux, Solaris). It is a complex environment that integrates many tools and allows working with many grids transparently and simultaneously. The main functionality concerns interactive grid application support, local and grid file management, security assurance, authorisation of access to resources and applications, and single sign-on technology based on X509 certificates.

1.1.3. JSS-Client installation

JSS-Client is an optional RPM with the client classes for Job Submission System.

It contains:

- the API's that developers of grid applications can use to implement their own JSS clients.;
- some programmes to test the services on RAS (see section 3.5.1)

It installs

- the JSS client jar library under the `/opt/cg/jss/share/java/client/ location;`
- some binaries for testing the JSS `(/opt/cg/jss/bin/cg-jss-*`);
- some example of jobs for the tests (`/opt/cg/jss/test`);
- the documentation (`/opt/cg/jss/docs`).

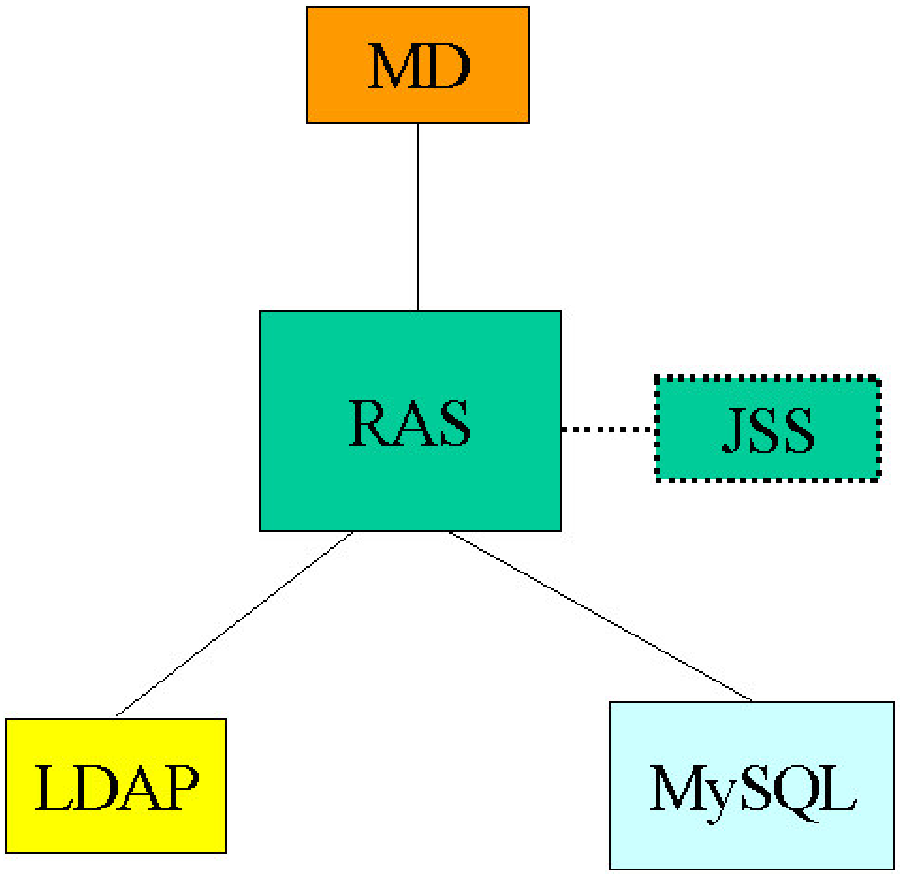
1.2. DEPENDENCIES

Package: WP3_1 Roaming Access and Migrating Desktop

The package consists of 4 parts:
- Roaming Access Server - wp3.1-RAS-services
- RAS LDAP database - wp3.1-RAS-ldap
- RAS MYSQL database - wp3.1-RAS-mysql

The Migrating Desktop - wp3.1-portals-MD is a “separate” product.

The diagram of dependency is shown on the picture.



Note that the placement of these components can be done on the single machine, or separately, one component per host respectively. JSS is incorporated into RAS installation package.

2. INSTALLATION IN THE CROSSGRID TESTBED

The CrossGrid testbeds are managed by the LCFG deployment support tool. This tool allows an automatic installation of the software on all required nodes.

2.1. RPM LISTS FOR LCFG

This section will be written in principle by the LCFG gurus from WP4. They know what to write here. The intention is to have that LCFG configuration documented somewhere. But it is the responsibility of the software developer to fill this paragraph by asking the LCFG gurus (or better pump them for this information).

2.2. PROFILE MODIFICATIONS FOR LCFG

This section will be written in principle by the LCFG gurus from WP4. They know what to write here. The intention is to have that LCFG configuration documented somewhere. But it is the responsibility of the software developer to fill this paragraph by asking the LCFG gurus (or better pump them for this information again and again and……).

2.3. MANUAL POST INSTALLATION STEPS

In most cases the installation will be fully automated, so you should tell here, that no extra post installation steps are required. If they are (like database setup or initialisation, special config-file), please write here what to do!

3. MANUAL INSTALLATION OF RAS

3.1. DOWNLOAD

RPM:
http://gridportal.fzk.de/distribution/crossgrid/autobuilt/i386-rh7.3-gcc3.2.2/wp3/RPMS/

Download the following files:

cg-wp3.1-RAS-ldap-*.noarch.rpm
cg-wp3.1-RAS-mysql-*.noarch.rpm
cg-wp3.1-RAS-services-*.noarch.rpm
(*) – choose the newest files. After development phase, the number will be fixed.

3.2. INSTALLATION FROM RPM

3.2.1. Prerequisites

cg-wp3.1-RAS-ldap-*.noarch.rpm requires:

- openldap >= 2.0.27-2.7.6multimaster
- openldap-servers >= 2.0.27-2.7.6multimaster
- openldap-clients >= 2.0.27-2.7.6multimaster
- sed
- /etc/openldap/slapd.conf

cg-wp3.1-RAS-mysql-*.noarch.rpm requires:

- MySQL-client >= 4.1.3
- MySQL-server >= 4.1.3

cg-wp3.1-RAS-services-*.noarch.rpm requires:

- tomcat4 >= 4.1.12
- cg-wp3.2-bypass_gcc3_2_2 >= 2.5.3-23
- cg-wp3.2-logging-api-cpp_gcc3_2_2 >= 2.1.15.2.2-1
- cg-wp3.2-chkpt-api_gcc3_2_2 >= 2.1.15.2.2-1
- cg-wp3.2-logging-api-sh_gcc3_2_2 >= 2.1.15.2.2-1
- cg-wp3.2-common-api_gcc3_2_2 >= 2.1.15.2.2-1
- cg-wp3.2-services-common_gcc3_2_2 >= 2.1.15.2.2-1
- cg-wp3.2-common-api-java_gcc3_2_2 >= 2.1.15.2.2-1
- cg-wp3.2-ui-api-cpp_gcc3_2_2 >= 2.1.15.2.2-1
- cg-wp3.2-common-api-java-interface_gcc3_2_2 >= 2.1.15.2.2-1

cg-wp3.2-ui-api-java_gcc3_2_2 >= 2.1.15.2.2-1
cg-wp3.2-config_gcc3_2_2 >= 2.1.15.2.2-1
cg-wp3.2-ui-api-java-interface_gcc3_2_2 >= 2.1.15.2.2-1
cg-wp3.2-interactive_gcc3_2_2 >= 2.1.15.2.2-1
cg-wp3.2-ui-cli_gcc3_2_2 >= 2.1.15.2.2-1
cg-wp3.2-jss-cpp-srv_gcc3_2_2 >= 2.1.15.2.2-1
cg-wp3.2-ui-config_gcc3_2_2 >= 2.1.15.2.2-1
lcg_util >= 1.4.1
j2sdk >= 1.4.2

The `cg-wp3.1-JSS-client-*.noarch.rpm` is an optional package and requires:

classads-jar = 1.1-2
cog-jar = 1.1-1
edg-java-security-client = 1.5.9-1
edg-java-security = 1.5.9-1
bouncycastle-jdk14 = 1.19-2

3.2.2. Users

Ensure you have the following user accounts BEFORE INSTALLATION on the RAS/MD machine:

apache
ldap
mysql
tomcat4

Every mentioned user must belong to the group with the same name as username. UIDs and GIDs are not important.
It is highly recommended to make these accounts from the LCFG level.

3.2.3. RAS-ldap installation

Install cg-wp3.1-RAS-ldap-*.noarch.rpm with LDAP-multimaster.

Example (download from: http://gridportal.fzk.de/distribution/crossgrid/releases/allfiles/7.3/cg/external/):
openldap-2.0.27-2.7.5multimaster.i386.rpm
openldap-clients-2.0.27-2.7.5multimaster.i386.rpm
openldap-devel-2.0.27-2.7.5multimaster.i386.rpm
openldap-servers-2.0.27-2.7.5multimaster.i386.rpm

The following two steps are optional:

– Synchronisation of LDAP content with the db snapshot received from RAS/MD team.
– Load applications received from RAS/MD team to LDAP.

If you are an administrator independent from CrossGrid, you are not obliged to contact with anybody.

3.2.4. LDAP settings for SSL

– Copy host key and certificate signed by CrossGrid CA from `/etc/grid-security/` to `/etc/openldap:`

```
chown ldap:ldap hostkey.pem
chown ldap:ldap hostcert.pem
```

– Change the owner of these files to `ldap:ldap`
– Uncomment the following in the configuration file: `/etc/openldap/slapd.conf`

```
TLSCertificateFile /etc/grid-security/hostcert.pem
TLSCertificateKeyFile /etc/openldap/hostkey.pem
TLSCACertificateFile /etc/grid-security/certificates/8a661490.0

(8a661490.0 is the example for Polish CA. Choose appropriate file.)
```

– `/etc/init.d/ldap restart`

3.2.5. /etc/hosts.allow

Modify `/etc/hosts.allow` in order to use ldap and Xvnc services. Ensure you have these two lines:

```
Slapd :      ALL
Xvnc  :      ALL
```

(Please note that on LCFG managed systems you need to modify the machine profile)

3.2.6. RAS-services installation

Modify `/etc/init.d/tomcat4`

```
  restart)
        stop
        sleep 10
        start
        sleep 5
        ;;
```

Sleep gives the time for proper shutdown of services before restarting.

Install cg-wp3.1-RAS-services-*.noarch.rpm

Set up the RAS configuration file: `/var/tomcat4/webapps/ras/config/RAS.conf`
Give the proper host addresses of services: SE, LDAP, JSS, FileManager, SQL. For the explanation of the file content see the chapter 3.4.2.

Configuring SSL over tomcat4:
Keystore:
```
keytool -genkey -keyalg RSA -alias tomcat -keystore ./keystore
```
In /var/tomcat4/conf/server.xml ensure you have the following uncommented:
```
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
               port="8443" minProcessors="5" maxProcessors="75"
               enableLookups="true"
               acceptCount="100" debug="0"
                           connectionTimeout= "20000" scheme="https" secure="true"
               useURIValidationHack="false" disableUploadTimeout="true">
      <Factory className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
               clientAuth="false" protocol="TLS"
               keystoreFile="/var/tomcat4/conf/keystore"
               keystorePass="desktop"/>
    </Connector>
```

`keystoreFile, keystorePass` – it depends on you, remember them from the previous (`keytool`) command. Check whether `keystoreFile` points to the right location.

Modify: `/var/tomcat4/webapps/ras/config/hostlistfile`:
```
.virtual_organisation_name https://ras_host_address:ssl_port
```
At the end of the `/var/tomcat4/conf/tomcat4.conf` file add :
```
export LD_LIBRARY_PATH=/opt/gcc-3.2.2/lib:/opt/globus/lib:/opt/edg/lib:/usr/local/lib:/usr/lib
export EDG_WL_LOCATION=/opt/edg
```

Restart tomcat4:
```
/etc/init.d/tomcat4 restart
```

3.2.7. MySQL installation

Prerequisites for MySQL with SSL installation:

MySQL settings for SSL:
To use SSL connections between the MySQL server and RAS client, your system must be able to support OpenSSL and your version of MySQL must be 4.1.1 or newer.

To get secure connections to work with MySQL, you must do the following:

1. Install the OpenSSL library. We have tested MySQL with OpenSSL 0.9.6. If you need OpenSSL, visit http://www.openssl.org.
2. Recompile MySQL. When you configure MySQL, run the configure script with the --with-vio and --with-openssl options. If you are using SRPM add --with-openssl option. You will have to remove also all-static flag.
3. Make sure that you have upgraded your grant tables to include the SSL-related columns in the mysql.user table. This is necessary if your grant tables date from a version prior to MySQL 4.0.0.

```
shell>mysql_fix_privilege_tables --password=root_password
```

4. To check whether a running mysqld server supports OpenSSL, examine the value of the have_openssl system variable:
5. `mysql> SHOW VARIABLES LIKE 'have_openssl';`

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_openssl  | YES   |
+---------------+-------+
```

If the value is YES, the server supports OpenSSL connections.

3.2.7.1. To run MySQL on SSL:

1. Create the CA (private key and public cert). (or use existing)
2. Create your server key. (or use existing)

Create server request and key

```
openssl req -new -keyout $DIR/server-key.pem -out \
        $DIR/server-req.pem -days 3600 -config $DIR/openssl.cnf
```

Remove the passphrase from the key (optional)

```
openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem
```

3. Create your server cert. (or use existing)

Sign server cert

```
openssl ca  -policy policy_anything -out $DIR/server-cert.pem \
    -config $DIR/openssl.cnf -infiles $DIR/server-req.pem
```

4. Modify/create `/etc/my.cnf` file and add following

```
[mysqld]
ssl-ca=$DIR/cacert.pem
```

```
ssl-cert=$DIR/server-cert.pem
ssl-key=$DIR/server-key.pem
```

Set the following properties in `RAS.conf` file, using the appropriate values for each property.

```
#RAS.conf file
#######################################################
#######  SQL Database settings              ##########
#######################################################

# server address
SQL.Server                = ras_host_address_put_here/

# do not change - connection params
SQL.ConnectionType        = jdbc:mysql://

# user and password
SQL.User                  = gridadmin
SQL.Passwd                = gridadmin

# name of Database and tables
SQL.Database              = GRID1
SQL.NodesTable            = nodes
SQL.UsersTable            = users

# with /without SSL support
SQL.SSL                   = true
```

3.2.7.2. SSL support

You will first need to import the MySQL server CA Certificate into a Java truststore. This is what SSL will use to determine if you are communicating with a secure MySQL server.

To use Java's 'keytool' to create a truststore in the current directory , and import the server's CA certificate ('cacert.pem'), you can do the following (assuming that'keytool' is in your path. It's located in the 'bin' subdirectory of your JDK or JRE):

```
keytool -import -alias mysqlServerCACert -file cacert.pem -keystore
truststore
```

You will then need to generate a client certificate, so that the MySQL server knows that it is talking to a secure client:

```
keytool -genkey -keyalg rsa -alias mysqlClientCertificate -keystore
keystore
```

Keytool will prompt you for the following information, and create a keystore named 'keystore' in the current directory.

Finally, to get RAS to use the keystore and truststore that you have generated, you need to set the following properties in RAS.conf file, replacing 'path_to_keystore_file' with the full path to the keystore file you created, 'path_to_truststore_file' with the path to the truststore file you created, and using the appropriate password values for each property.

```
RAS.conf file:

[SQL]
SSL=true

[SSL]
keyStore= 'path_to_keystore_file'
keyStorePassword=****
trustStore='path_to_truststore_file'
trustStorePassword=******
```

End of prerequisites.

3.2.8. RAS-MySQL Installation

Install cg-wp3.1-RAS-mysql-*.noarch.rpm

Create new MySQL db for RAS (give the MySQL root's password or hit Enter if the password hadn't been set before).

```
mysql -u root -p < /opt/cg/etc/mysql/crossgrid.sql
mysql -u root -p < /opt/cg/etc/mysql/crossgrid_update.sql
```

3.3. INSTALLATION FROM SOURCE

Download sources from:

http://gridportal.fzk.de/distribution/crossgrid/autobuilt/i386-rh7.3-gcc3.2.2/wp3/SOURCES/

cg-wp3.1-RAS-ldap-*.tar.gz

cg-wp3.1-RAS-mysql-*.tar.gz

cg-wp3.1-RAS-services-*.tar.gz
cg-wp3.1-JSS-client-*.tar.gz

You need the Ant compiler preinstalled. Proper target invokes compilation and building rpms:

```
ant rpm-ldap
ant rpm-mysql
ant rpm-ras
ant rpm-jssclt
```

You will get rpms ready to install.

3.4. CONFIGURATION

3.4.1. List of configuration files

```
/var/tomcat4/webapps/ras/config/RAS.conf
/var/tomcat4/webapps/ras/config/log4j.conf
/var/tomcat4/webapps/ras/config/crossgrid.conf
/var/tomcat4/webapps/ras/config/hostlistfile
/var/tomcat4/webapps/ras/plugins/plugin.properties
```

Following two files are installed by Tomcat4 but also needs some modification for RAS:

```
/etc/init.d/tomcat4
/var/tomcat4/tomcat.conf
```

3.4.2. Editing the configuration files

It is described above. See 3.2.
Validate the plugin addresses in file:
`/var/tomcat4/webapps/ras/plugins/plugin.properties`
Validate RB and LB address in
`/var/tomcat4/webapps/ras/config/crossgrid.conf`

Example of:

```
#/var/tomcat4/webapps/ras/config/log4j.conf
#
log4j.rootLogger=ERROR, A1
log4j.category.org.crossgrid.wp3=DEBUG
log4j.category.org.apache.axis=OFF
log4j.category.org.crossgrid.wp3.portals.roamingaccessserver.ldapmanager=OFF
```

```
log4j.category.org.globus=WARN

log4j.appender.A1=org.apache.log4j.ConsoleAppender

#log4j.appender.A1=org.apache.log4j.RollingFileAppender
#log4j.appender.A1.File=/var/tomcat4/webapps/ras/log/RAS.log
#log4j.appender.A1.MaxFileSize=100KB
# Keep one backup file
#log4j.appender.A1.MaxBackupIndex=20

log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=---%5p--- %c %x - %m%n
```

Example of:

```
#/var/tomcat4/webapps/ras/config/RAS.conf
#if CA certificates are not in getCaCertLocations set
#CoG.Properties.Path to cog.properties file which define
#cacert property
#CoG.Properties.Path =

###################################################
###############   SE Config   ####################
###################################################
#Storage Element section. All values should be set.

#Mode of connection between Client and SE {passive|active}
SE.Mode = passive

###################################################
###############   LDAP Config   ##################
###################################################

#Host address with the installation of LDAP db.
LDAP.Host = willow.crossgrid.man.poznan.pl

#LDAP Domain name:
LDAP.DN   = dc=ras, dc=crossgrid, dc=org

# true - connect to LDAP over SSL (port 636)
# false - connect to LDAP without SSL (port 389)
LDAP.SSL   = false

#Common name of user:
LDAP.User = cn=root, dc=ras, dc=crossgrid, dc=org

#LDAP password goes here in plain text:
LDAP.Passwd = secret


###################################################
```

```
###############   JSS Config   ####################
###################################################
#Job Submission Service (typically on the same machine as RAS)
#protocol name:
JSS.Protocol = http
#host where the JSS service is installed
JSS.Host     = ras.man.poznan.pl
#port number for JSS:
JSS.Port     = 8080
#path to JSS service
JSS.Path     = /ras/services/JobSubmission


###################################################
###############   RB Config   #####################
###################################################
#these data about Resource Broker and Loggingandbookkeeping you must get from administrator of
the particular service.
#Resource Broker host and port (default):
RB.Host = rb01.lip.pt
RB.Port = 7772

#Loggingandbookkeeping host and port (default):
LB.Host = rb01.lip.pt
LB.Port = 9000

###################################################
###############   FileManager Config   ############
###################################################
# The address of FileManagerService of RAS. Set it to your RAS server unless you have really
special needs.

#protocol
FMS.Protocol = https
#hostaddress of your ras installatiton.
FMS.Host     = ras.man.poznan.pl
#Service port (default)
FMS.Port     = 8443
#Path to the service
FMS.Path     = /ras/services/FileManagerWSServicePortType


######################################################
#######  SQL Database settings              ##########
######################################################
#host address of SQL dbms
SQL.Server              = willow.crossgrid.man.poznan.pl/
#currently MySQL driver is used - do not change unless you know what you are doing:
SQL.ConnectionType      = jdbc:mysql://
#data for accessing db:
SQL.User                = gridadmin
SQL.Passwd              = gridadmin
SQL.Database            = GRID1
SQL.NodesTable          = nodes
SQL.UsersTable          = users
SQL.SSL               = false



####################################################
###############   II Config   ###################
####################################################
#Information index
#Ask II administrator for these data:
II.Host = ii01.lip.pt:2170
#II.Host = ic.fzk.de:2170
II.DN   = Mds-vo-name=local,o=grid



####################################################
###############   VO Config   ###################
####################################################
#Virtual Organisation

#VO name. Currently, one instance of RAS supports one VO. Other names in Crossgrid are cg-tut, cg-test
VO = cg

#Choose the location of hostlist_file and grid-mapfile:
HOSTLIST_FILE = /var/tomcat4/webapps/ras/config/hostlistfile
GRID_MAPFILE = /etc/grid-security/grid-mapfile



####################################################
###############   SQL/SSL Config   ###################
####################################################

# Fill in when you configure MySQL with SSL:

#path_to_keystore_file
SSL.keyStore=

#put the password here:
SSL.keyStorePassword=

#path to the truststore file:
SSL.trustStore=

#put the password for truststore file here:
SSL.trustStorePassword=

####################################################
```

```
###############   RM/LCG UTILS Config   ###########
##################################################

LCGUTILS.LD_LIBRARY_PATH = /opt/lcg/lib
LCGUTILS.GFAL_INFOSYS = ii01.lip.pt:2170
LCGUTILS.LCG_LOCATION = /opt/lcg
LCGUTILS.VAR = /opt/lcg/var
LCGUTILS.SYSCONFIG = /etc/sysconfig/lcg
LCGUTILS.TMP = /tmp
```

3.4.3. Startup scripts

N/A

3.4.4. Other requirements

N/A

3.4.4.1. Environment

N/A

3.4.4.2. Users

Ensure you have the following user accounts BEFORE INSTALLATION on the RAS machine:

apache
ldap
mysql
tomcat4

Every mentioned user must belong to the group with the same name as username.
UIDs and GIDs are not important.

3.4.4.3. Ports

N/A

Remark:

Ensure to have open ports 8443, 2811, and poll 13000-17000 to Migrating Desktop client.

3.4.4.4. Certificates

Certificates are needed for ldap, mysql and tomcat. You can use existing machine certificate or generate them with keytool.

We recommend using existing machine certificate for ldap and generate java keystores for mysql and tomcat4. If you would like to use other method, consult individual product configuration manual.

3.4.4.5. Folders

3.5. RUNNING AND TESTING

```
/etc/init.d/tomcat4 {start|stop}
```

The best way is to have installed Migrating Desktop and test the availability of services from that client.

3.5.1. JSS Testing

The `cg-wp3.1-JSS-client-*.noarch.rpm` package (see section 3.2) provides some test programmes `(/opt/cg/jss/bin/cg-jss-*)` and some examples of test `(/opt/cg/jss/test).`

The test programs are the following:

1) jsstest
2) jsscredential
3) jobsubmit
4) jobstatus
5) loginfo
6) jobcancel
7) listmatch
8) userjobs
9) interactivity

Except for the first test (`cg-jss-webservices-test`), the tester has to use a valid X509 proxy. All the programmes allow users to set the path location of the proxy file as optional input parameter.If no proxy file path is specified, it will be checked the X509_USER_PROXY environment variable; if this variable isn't set, the default /tmp/x509_u<uid> file will be used.

3.5.1.1. A simple test.

It allows checking whether the JSS are running on the specified web server without using any X509 proxy.

```
cg-jss-webservices-test <WS_URL> <MESSAGE>
```

where:

WS_URL= the Web Services URL (mandatory)

MESSAGE= a short text message that is sent to the web server (mandatory)

The service just makes an "echo" of the string message .

For instance:

cg-jss-webservices-test http://cedar.man.poznan.pl:8080/ras/services/JobSubmission crossgrid

or

cg-jss-webservices-test https://cedar.man.poznan.pl:8443/ras/services/JobSubmission crossgrid

3.5.1.2. Testing of the user credential.

It allows to check if the JSS are running on the specified web server.

```
        cg-jss-credential-test <WS_URL> <MESSAGE> [<PROXY_FILE>]
```

where:

<WS_URL> = the Web Services URL (mandatory)

<MESSAGE> = a short text message that is sent to the web server (mandatory)

<PROXY_FILE>= path location to a X509 proxy file (optional)

For instance:

cg-jss-credential-test http://cedar.man.poznan.pl:8080/ras/services/JobSubmission crossgrid /tmp/x509up_example [1]

or

cg-jss-credential-test https://cedar.man.poznan.pl:8443/ras/services/JobSubmission crossgrid /tmp/x509up_example [2]

3.5.1.3. Submission

It allows submitting a job to the specified Resource Broker (Network Server).

```
cg-jss-jobsubmit-test <input-file> <JDL-File>
```

where:

<input-file> = the location of the file containing the info needed to the service

<JDL-file> = the location of the file containing the info needed to the service

#///

# TEMPLATE for THE input-File

#///

```
#Replace the string "< >" with the proper values
```

[1] The proxy file parameter is optional

[2] The proxy file parameter is optional

```
#
#       # Web Services URL
#       WS_URL=<WebServices-URL=the web services URL>
#
#       # RB hostname
#       RB_HOST=<LB-Host=the hostname where the RB is running>
#
#       # RB port
#       RB_PORT=<LB-Port= the port number used to contact the RB>
#
#       # L&B hostname
#       LB_HOST=<LB-Host=the hostname where the LB is running>
#
#       # L&B port
#       LB_PORT=<LB-Port= the port number used to contact the RB>

#   Computing Element ID (to force submission) NOT MANDATORY!!!!
#   CE_ID=<full-hostname>:<port-number>/jobmanager-<service>-<queuename>
#     CE_ID=
#
#   PROXY_FILE=<Proxy=path location to proxy file> (not mandatory !!)
#

#//////////////////////////////////////////////////////////////////////////
/
#end TEMPLATE
#//////////////////////////////////////////////////////////////////////////
/
```

For instance, the input file :

```
# Web Services URL
WS_URL=https://cedar.man.poznan.pl:8443/ras/services/JobSubmission
# RB hostname
RB_HOST=rb.fzk.de
# RB port
RB_PORT=7772
# L&B hostname
LB_HOST=rb01.lip.pt
# L&B port
LB_PORT=9000
# CEId
CE_ID=zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-long
```

```
# proxy
PROXY_FILE=
```

The JDL file:

```
Executable="/bin/ls" ;
StdOutput       = "std.out";
StdError        = "std.err";
OutputSandbox   = {"std.out","std.err"};
VirtualOrganisation = "cg";
RetryCount = 3 ;
requirements =  other.GlueCEStateStatus == "Production" ;
rank =  other.GlueCEStateEstimatedResponseTime;
```

The command is:

```
cg-jss-jobsubmit-test /opt/cg/jss/test/etc/submit_example.in \ /opt/cg/jss/test/jdl/ex1.jdl
```

3.5.1.4. Job Status

It allows retrieving the status of a submitted job.

```
cg-jss-jobstatus-test <WS_URL> <JOBID> [<PROXY_FILE>]
```

where:

<WS_URL> = the Web Services URL (mandatory)

<JOBID> = the identifier of the job (mandatory)

<PROXY_FILE>= path location to a X509 proxy file (optional; if no path is specified, the default /tmp/x509_u<uid> file is loaded)

for instance:

```
cg-jss-jobstatus-test  http://cedar.man.poznan.pl:8080/ras/services/JobSubmission \
https://rb01.lip.pt:9000/X__6HnFN51PdCwjnFjDFvQ
```

or

```
cg-jss-jobstatus-test  https://cedar.man.poznan.pl:8443/ras/services/JobSubmission \
https://rb01.lip.pt:9000/X__6HnFN51PdCwjnFjDFvQ
```

3.5.1.5. Job Log Info,

It allows retrieving the logging information of a submitted job

```
cg-jss-loginfo-test <WS_URL> <JOBID> [<PROXY_FILE>]
```

where:

<WS_URL> = the Web Services URL (mandatory)

<JOBID> = the identifier of the job (mandatory)

<PROXY_FILE>= path location to a X509 proxy file (optional; if no path is specified, the default /tmp/x509_u<uid> file is loaded)

For instance:

cg-jss-loginfo-test http://cedar.man.poznan.pl:8080/ras/services/JobSubmission \
https://rb01.lip.pt:9000/X__6HnFN51PdCwjnFjDFvQ

or

cg-jss-loginfo-test https://cedar.man.poznan.pl:8443/ras/services/JobSubmission \
https://rb01.lip.pt:9000/X__6HnFN51PdCwjnFjDFvQ

3.5.1.6. Job Cancel

It allows cancelling a previous submitted job. As it was previously written, it is just a request of cancelling: it's sent to the RB and if the status of the job allows the operation (for example the status is different from "ABORTED" or "DONE"), the job will be cancelled.

The syntax of the command:

```
cg-jss-jobcancel-test <WS_URL> <JOBID> [<PROXY_FILE>]
```

where:

<WS_URL> = the Web Services URL (mandatory)

<MESSAGE> = a short text message that is sent to the web server (mandatory)

<PROXY_FILE>= path location to a X509 proxy file (optional; if no path is specified, the default /tmp/x509_u<uid> file is loaded)

for instance:

cg-jss-jobcancel-test http://cedar.man.poznan.pl:8080/ras/services/JobSubmission \
https://rb01.lip.pt:9000/Tpto9pH8dF5wm8TMWwfopg

or

cg-jss-jobcancel-test https://cedar.man.poznan.pl:8443/ras/services/JobSubmission \
https://rb01.lip.pt:9000/Tpto9pH8dF5wm8TMWwfopg

3.5.1.7. Job List Match

It performs the match-making process on the specified RB (Network Server),
in order to know the CE matching with the requirements in the JDL.

```
cg-jss-listmatch-test  <WS-URL > <RB-Host> <RB-Port> <JDL_file> [<PROXY_FILE>]
```

where:

<WS_URL> = the Web Services URL (mandatory)

<RB_HOST> = the hostname where the RB is running (mandatory)

<RB_PORT> = the port number where the RB is running (mandatory)

<JDL_file>=path location to the JDL file (mandatory)

<PROXY_FILE>= path location to a X509 proxy file (optional; if no path is specified, the default /tmp/x509_u<uid> file is loaded)

for instance:

```
cg-jss-listmatch-test \
http://cedar.man.poznan.pl:8080/ras/services/JobSubmission rb01.lip.pt 7772 /opt/cg/jss/test/jdl/ex1.jdl
```

or

```
cg-jss-listmatch-test \
https://cedar.man.poznan.pl:8443/ras/services/JobSubmission rb01.lip.pt 7772 \
/opt/cg/jss/test/jdl/ex1.jdl
```

3.5.1.8. User Jobs

It allows retrieving a list of the previous submitted jobs from the specified LB.

```
cg-jss-userjobs-test <WS_URL> <LB_Host> <LB_Port> [<PROXY_FILE>]
```

<WS_URL> = the Web Services URL (mandatory)

<LB_HOST>= the hostname where the LB is running (mandatory)

<LB_PORT>= the port number where the LB is running (mandatory)

<PROXY_FILE>= path location to a X509 proxy file (optional; if no path is specified, the default /tmp/x509_u<uid> file is loaded)

For example:

```
cg-jss-userjobs-test \
http://cedar.man.poznan.pl:8090/ras/services/JobSubmission rb01.lip.pt 9000 /tmp/x509up_example
```

or

```
cg-jss-userjobs-test \
https://cedar.man.poznan.pl:8443/ras/services/JobSubmission rb01.lip.pt 9000 /tmp/x509up_example
```

3.5.1.9. *Interactivity*

It allows submitting interactive jobs and launching a console to handle interactive sessions. In order to perform this test firewall has to be set in order to accept inbound connection coming from the CE where the job will be sent. A range of ports to use for Job Shadow can be defined by setting the environment variable GLOBUS_TCP_PORT_RANGE:

The value of this variable has to be formatted as:

```
GLOBUS_TCP_PORT_RANGE=”<val min> <val max>”
```

For example:

```
export GLOBUS_TCP_PORT_RANGE=”34000 35000”
```

In order to test interactive services, there are two testing programs that needed to be launched.

The first program is :

```
cg-jss-listener-test <unique_string> <nodes_number>
```

`<unique_string>=` a unique string that is used to identifier the started instance (mandatory)

`<nodes_number>=`the number of nodes required to run a MPICH-G2 jobs (mandatory only for MPICH-G2 jobs; for any other kind of job don’t specify anything)

It allows starting:

1. a JobShadow instance, looking for a free port in the range specified by the GLOBUS_TCP_PORT_RANGE variable (making a scanning of the ports, starting from the ;
2. a Listener Console where, during the interactive session, the job output will be shown and the tester can type and send to the job the input data

The Job Shadow writes a set of information (hostname and port number where it’s running; the name of the i/opipes; its process id number) into the file:

```
/tmp/cg-jss-SHADOW-<unique_string>.info
```

For Example :

```
[cg_dev@cgrid.datamat.it] cg-jss-listener-test mytest
Unable to use port 34000! (Address already in use)
Unable to use port 34001! (Address already in use)
Unable to use port 34002! (Address already in use)
Unable to use port 34003! (Address already in use)
JobShadow information
------------------------------------------------------
```

```
HOST         = [150.254.173.164]
PORT         = [34004]
PID          = [26516]
PIPES        = [/tmp/listener-LXcjNR7P2hYlPd4W_n2DPg]
-------------------------------------------------------
*********************************************************************
***************  Interactive Session Console ********************
*********************************************************************
```

Now the executable that takes care to submit the interactive job has to be launch:

```
cg-jss-interactivity-test <input-file> <JDL-File> <ShadowInfo-File>
```

<input-file> = the location of the file containing the info needed to the service

<JDL-file> = the location of the file containing the info needed to the service

<unique-string>=the unique string that was used to identifier the JobShadow

It needs to read three files:

1. the input file containing the information on Web Services URL, RB, L&B etc… (it has the same structure as the file used for submission of normal job; see section **Błąd! Nie można odnaleźć źródła odsyłacza.**) ;
2. the JDL file ;
3. the location path of the file produced by the Job Shadow previously started (/tmp/cg-jss-SHADOW-<unique_string>.info).

If the job is successfully submitted, the cg-jss-interactivity-test programs will get back the jobid assigned by Submission service to the job.

When the job start running, the listener will receive the messages coming from the CE.

For instance, the input file :

```
WS_URL=http://cedar.man.poznan.pl:8090/ras/services/JobSubmission
#NetworkServer hostname
NS_HOST=rb02.lip.pt
#NetworkServer hostname
NS_PORT=7772
# L&B hostname
LB_HOST=rb02.lip.pt
# L&B port
LB_PORT=9000
```

```
# CE_ID
CE_ID=zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-infinite
# proxy-file
PROXY_FILE=
```

The RPM installation provides some jobs as example under the `/opt/cg/jss/test/jobs` directory:

- simple (that contains containing some simple examples to test one-way and two-way interactivity);
- mpichg2 (to test MPICH-G2 interactive jobs);
- mpichp4 (to test MPICH-P4 interactive jobs);

Hereafter we show a test that was performed using the "*gas*" program that is distributed among the "simple-job"-examples.

We launch the listener:

```
cg-jss-listener-test mytest
```

After that the listener is successfully started, we can submit our job (specifying the file that containing the information on the launched JobShadow as the third parameter):

```
cg-jss-interactivity-test interactivity.in interactivity.jdl \
/tmp/cg-jss-SHADOW-mytest.info
```

```
TEST : JobSubmit-Service
********************************************************************************
WS URL        = [http://cedar.man.poznan.pl:8090/ras/services/JobSubmission]
---------------------------------------------------------------------------------------------------------------------
certificate:
************************************************************
Default proxy file.
- Getting default credential
Cred    issuer:    /C=IT/O=INFN/OU=Personal    Certificate/L=DATAMAT    DSAGRD/CN=Marco
Sottilaro/Email=marco.sottilaro@datamat.it
Time Left      : 43132 sec ( 00 days : 11 hours : 58 min : 52 sec)
---------------------------------------------------------------------------------------------------------------------
NS host        = [rb02.lip.pt]
NS port        = [7772]
---------------------------------------------------------------------------------------------------------------------
LB host        = [rb02.lip.pt]
LB port        = [9000]
---------------------------------------------------------------------------------------------------------------------
ceId           = [zeus24.cyf-kr.edu.pl:2119/jobmanager-pbs-infinite]
```

```
-----------------------------------------------------------------------------------------------------------------------
JDL                  = [[  Executable = "test-gas" ; VirtualOrganisation = "cg" ; InputSandbox =
"/home/portal_test/Installation/opt/cg/jss/test/gas/test-gas" ; RetryCount = 3 ; requirements =
other.GlueCEStateStatus == "Production" ; rank = other.GlueCEStateEstimatedResponseTime ;
JobType = "interactive" ; JobType = "interactive" ;  ]]
-----------------------------------------------------------------------------------------------------------------------
************************************************************************
      INTERACTIVE JOB
      ---------------
************************************************************************
SHADOW host          = [150.254.173.164]
SHADOW port          = [34006]
SHADOW pipes          = [/tmp/listener-LXcjNR7P2hYlPd4W_n2DPg]
---------------------------------------------
Connecting to [http://cedar.man.poznan.pl:8090/ras/services/JobSubmission]
jobSubmit [end]
jobid: https://rb02.lip.pt: 9000/pGEj2IVOoB9mwT5RQxxijA
TEST JOB-INTERACTIVITY: ok
```

When the job starts running, the output messages of the job will show on the Listener Console.

```
[cg_dev@cgrid.datamat.it] cg-jss-listener-test mytest
Unable to use port 34000! (Address already in use)
Unable to use port 34001! (Address already in use)
Unable to use port 34002! (Address already in use)
Unable to use port 34003! (Address already in use)
JobShadow information
------------------------------------------------------
HOST           = [150.254.173.164]
PORT           = [34004]
PID            = [26516]
PIPES          = [/tmp/listener-LXcjNR7P2hYlPd4W_n2DPg]
------------------------------------------------------
**********************************************************************
***************  Interactive Session Console ********************
**********************************************************************
Type $QUIT to close the session
------------------------------------------------------------------------
THE IDEAL-GAS EQUATION
==========================================
1. pressure
2. volume
3. moles
```

```
4. temperature
Choose the unknown quantity [1-4]: 1
<1>
Insert volume value (L): 2
Insert moles value: 3
Insert temperature value: 4


THE PRESSURE VALUE IS: 0.492 atm
Do you wish continue [y/n]? n
Bye!

$QUIT
Removing pipes … [ok]
Stopping ….[ok]
Interactive Session end
```

An example of test for MPICH-G2 is shown. In the RPM distribution the source for this test is not built: it has to be done using the compiler dedicated to MPICH-G2 jobs. Installing the `ch-g2-x.y.z.j-cg2` package, it is `/opt/mpi/mpich-g2/bin/mpicc.`

In order to specify this location, we need to edit the Makefile and set the `MPICH_G2_INSTALL_PATH` variable.

We launch the testing program for the listener, specifying in the second parameters the number of nodes that we need (in this example we need two nodes):

```
cg-jss-listener-test mytest 2
```

After that the listener is successfully started, we can submit our job (specifying the file that containing the information on the launched JobShadow as the third parameter):

```
cg-jss-interactivity-test interactivity.in interactivity.jdl \
/tmp/cg-jss-SHADOW-mytest.info
```

Hereafter an example of the test:

```
[portal_test@cedar sbin]$ cg-jss-listener-test mytest 2
EDG location is /home/portal_test/Installation/opt/edg
JobShadow information
--------------------------------------------------------
HOST    = [150.254.173.164]
PORT    = [34000]
PID     = [16021]
PIPES   = [/tmp/listener-A8adScaWkFGiMlldd0iSVQ]
```

```
-------------------------------------------------------
NUMBER OF NODES (only for MPICH-G2 jobs)        = [2]
****************************************************************
***************  Interactive Session Console *******************
****************************************************************
------------------------------------------------------------------------
Type $QUIT to close the session
------------------------------------------------------------------------
my_id 0 numprocs 2
Number of trips around the ring: 4
Verbosity (yes/no): yes
Processor name: zeus32.cyf-kr.edu.pl
Starting trip 1 of 4: before sending num=1 to dest=1
Inside trip 1 of 4: before receiving from source=1
End of trip 1 of 4: after receiving passed_num=2 (should be =trip*numprocs=2) from source=1
Starting trip 2 of 4: before sending num=3 to dest=1
Inside trip 2 of 4: before receiving from source=1
End of trip 2 of 4: after receiving passed_num=4 (should be =trip*numprocs=4) from source=1
Starting trip 3 of 4: before sending num=5 to dest=1
Inside trip 3 of 4: before receiving from source=1
End of trip 3 of 4: after receiving passed_num=6 (should be =trip*numprocs=6) from source=1
Starting trip 4 of 4: before sending num=7 to dest=1
Inside trip 4 of 4: before receiving from source=1
End of trip 4 of 4: after receiving passed_num=8 (should be =trip*numprocs=8) from source=1
>>>>>>>>>>>>>>>  INTERACTIVE JOB FINISHED  <<<<<<<<<<<<<<<
my_id 1 numprocs 2
Processor name: zeus16.cyf-kr.edu.pl
Top of trip 1 of 4: before receiving from source=0
Inside trip 1 of 4: after receiving passed_num=1 from source=0
Inside trip 1 of 4: before sending passed_num=2 to dest=0
Bottom of trip 1 of 4: after send to dest=0
Top of trip 2 of 4: before receiving from source=0
Inside trip 2 of 4: after receiving passed_num=3 from source=0
Inside trip 2 of 4: before sending passed_num=4 to dest=0
Bottom of trip 2 of 4: after send to dest=0
Top of trip 3 of 4: before receiving from source=0
Inside trip 3 of 4: after receiving passed_num=5 from source=0
Inside trip 3 of 4: before sending passed_num=6 to dest=0
Bottom of trip 3 of 4: after send to dest=0
Top of trip 4 of 4: before receiving from source=0
Inside trip 4 of 4: after receiving passed_num=7 from source=0
Inside trip 4 of 4: before sending passed_num=8 to dest=0
```

```
Bottom of trip 4 of 4: after send to dest=0
>>>>>>>>>>>>>>>  INTERACTIVE JOB FINISHED  <<<<<<<<<<<<<<<
```

Now a test for MPICH-P4 is shown. In the RPM distribution the source for this test is not built: it has to be done using the compiler dedicated to MPICH jobs. In order to specify this location, we need to edit the Makefile and set the `MPICH_P4_INSTALL_PATH` variable.

We launch the testing program for the listener without specifying the number of nodes as MPICH-G2 jobs (see the previous section):

```
cg-jss-listener-test mytest
```

After that the listener is successfully started, we can submit our job (specifying the file that containing the information on the launched JobShadow as the third parameter):

```
cg-jss-interactivity-test interactivity.in interactivity.jdl \
/tmp/cg-jss-SHADOW-mytest.info
```

Hereafter an example of the test:

```
cg-jss-listener-test mytest
EDG location is /home/portal_test/Installation/opt/edg
JobShadow information
-------------------------------------------------------
HOST    = [150.254.173.164]
PORT    = [34000]
PID     = [16494]
PIPES   = [/tmp/listener-Dsxd9HAxc0ZnSVWUcqSY-Q]
-------------------------------------------------------
NUMBER OF NODES (only for MPICH-G2 jobs)        = [0]
****************************************************************
***************  Interactive Session Console ******************
****************************************************************
-------------------------------------------------------------------------
Type $QUIT to close the session
-------------------------------------------------------------------------
my_id 0 numprocs 2
Number of trips around the ring: 3
Verbosity (yes/no): yes
my_id 1 numprocs 2
Slave 1: Processor name: zeus33.cyf-kr.edu.pl
Slave 1: top of trip 1 of 3: before receiving from source=0
Slave 1: inside trip 1 of 3: after receiving passed_num=1 from source=0
Slave 1: inside trip 1 of 3: before sending passed_num=2 to dest=0
```

```
        Slave 1: bottom of trip 1 of 3: after send to dest=0
        Slave 1: top of trip 2 of 3: before receiving from source=0
        Slave 1: inside trip 2 of 3: after receiving passed_num=3 from source=0
        Slave 1: inside trip 2 of 3: before sending passed_num=4 to dest=0
        Slave 1: bottom of trip 2 of 3: after send to dest=0
        Slave 1: top of trip 3 of 3: before receiving from source=0
        Slave 1: inside trip 3 of 3: after receiving passed_num=5 from source=0
        Slave 1: inside trip 3 of 3: before sending passed_num=6 to dest=0
        Slave 1: bottom of trip 3 of 3: after send to dest=0
        Master: Processor name: zeus33.cyf-kr.edu.pl
        Master: starting trip 1 of 3: before sending num=1 to dest=1
        Master: inside trip 1 of 3: before receiving from source=1
        Master: end of trip 1 of 3: after receiving passed_num=2 (should be =trip*numprocs=2) from
        source=1
        Master: starting trip 2 of 3: before sending num=3 to dest=1
        Master: inside trip 2 of 3: before receiving from source=1
        Master: end of trip 2 of 3: after receiving passed_num=4 (should be =trip*numprocs=4) from
        source=1
        Master: starting trip 3 of 3: before sending num=5 to dest=1
        Master: inside trip 3 of 3: before receiving from source=1
        Master: end of trip 3 of 3: after receiving passed_num=6 (should be =trip*numprocs=6) from
        source=1
>>>>>>>>>>>>>>>  INTERACTIVE JOB FINISHED  <<<<<<<<<<<<<<<
```

3.5.2. log files

```
/var/tomcat4/logs/catalina.out
/tmp/cg-jss-tomcat4-<date>.log
```

4. MANUAL INSTALLATION OF MD

4.1. DOWNLOAD

RPM:
http://gridportal.fzk.de/distribution/crossgrid/autobuilt/i386-rh7.3-gcc3.2.2/wp3/RPMS/

4.2. INSTALLATION FROM RPM

Download and install the following files:
cg-wp3.1-portals-MD-*.noarch.rpm
(*) – choose the newest files. After development phase, the number will be fixed.

Ensure, you have web server running (e.g. Apache).

4.3. INSTALLATION FROM SOURCE

Download sources from:
http://gridportal.fzk.de/distribution/crossgrid/autobuilt/i386-rh7.3-gcc3.2.2/wp3/SOURCES/

cg-wp3.1-portals-MD-*.tar.gz

You need the Ant compiler preinstalled. Proper target invokes compilation and building rpms:

```
ant rpm-md
```

You will get rpm ready to install.

4.4. CONFIGURATION

4.4.1. List of configuration files

4.4.2. Editing the configuration files

4.4.3. Startup scripts

4.4.4. Other requirements

4.4.4.1. Environment

N/A

4.4.4.2. Users

Root privileges (in order to install from rpm).

4.4.4.3. Ports

Open the firewalls for 8443 port. If you want to transfer files you should keep open the pool: 13000-17000 (If it is not possible please always use tunnelled connection in MD) and 2811 in both directions between your local workstation and remote SE(s).

4.4.4.4. Certificates

The user credentials shoud be valid. Create user's certificate using CrossGrid UI machine

```
>globus-cert-request
```

and folow the instructions. To sign certificate contact your country CA:

Certification Authorities used for CrossGrid Testbed Sites. Please ensure the proper CA for your country.

4.4.4.5. Folders

4.5. RUNNING AND TESTING

There are two different ways of launching Migrating Desktop: using Java Web Start or Web Browser. We strongly recommend to use Java Web Start.

If you prefer Java Web Start
Requirements:

- Install JWS 1.4.2 or newer.
- Run MD from this location: http://<installation_host>/crossgrid/JWS/MigratingDesktop.jnlp
- Problems(*) encountered during loading MD under SUN JWS 1.4.2:

  - If you see "Security Warning" - use START button;
  - If you see next warning similar to "failed to verify certificate" do not use CLOSE button - use window upper right "x" button;

  * Remark: JWS has a problem, not MD.

Remember the proper RAS address:

- Fill in the RAS server field: **RAS_HOSTNAME** and port **8443**
- You will be asked for your x509 certificate (you can save the settings for future use).

You can check, whether the installation works using the following services:

File transferring via Grid Commander.

Submitting the job e.g. crossgrid commandline.

Checking the status of the job via Job Monitoring tool.

Store/restore user profile.

4.5.1. log files

See log files of Apache and java console.

5. THE CROSSGRID LICENSE AGREEMENT

6. BIBLIOGRAPHY