Project number: RI 211727

Project acronym: EDGeS

Project full title: Enabling Desktop Grids for e-Science

Research Infrastructures
INFRA-2007-1.2.2 Deployment of eInfrastructures for scientific communities

DSA1.1
EGEE Application Repository and a Desktop Grid EGEE VO

Due date of deliverable: 30/09/2008 Actual submission date: 30/09/2008

Start date of project: 01/01/2008 Duration: 24 months

MTA SZTAKI

Dissemination level: PU

Version: 1.5

The EDGeS project has received Community research funding

1 Table of Contents

2 Status and Change History

Status:	Name:	Date:	Signature:
Draft:	Zoltán Balaton	18/09/2008	n.n. electronically
Reviewed:	Raúl Ramos	23/09/2008	n.n. electronically
Approved:	Péter Kacsuk	30/09/2008	n.n. electronically

Version	Date	Pages	Authors	Modification
1.0	10/09/2008	all	KG	EDGeS AR text added
1.1	15/09/2008	all	BZ	revision of the deliverable template
1.2	15/09/2008	all	KG	added figures
1.3	15/09/2008	all	BZ	Desktopgrid VO text added
1.4	25/09/2008	all	BZ	implement reviewer's comments, final formatting
1.5	29/09/2008	all	BZ	final version

3 Glossary

AR	Application Repository
BOINC	Berkeley Open Infrastructure for Network Computing
CLI	Command Line Interface
CE	Computing Element
CRUD	Create, Read, Update, Delete
DG	Desktop Grids: Grids composed of Desktop PCs.
EDGeS	Enabling Desktop Grids for E-Science, the FP7 project that aims to bridge Service Grids and Desktop Grids
EGEE	The Enabling Grids for E-sciencE (EGEE) project is funded by the European Commission and aims to build on recent advances in grid technology
GEMLCA	Grid Execution Management for Legacy Code Applications
JDL	Job Description Language
PKI	Public Key Infrastructure
SG	Service Grids: Grids composed of resources belonging to institutions.
VO	Virtual Organisation
VOMS	gLite VO Membership Service
WMS	gLite Workload Management System

4 Introduction

The objective of this deliverable is to describe the Application Repository Service based on GEMLCA which implements the conceptual Application Repository that stores validated, certified applications. The other entity described here is the desktopgrid VO that is created to collect EGEE resources which will accept jobs from bridged desktop grids. The integrated EDGeS infrastructure using these components is described in a separate deliverable (DSA1.2).

5 Application repository usage scenarios

Depending on who uses the repository we define user, site and bridge level scenarios. These are described in the following.

5.1 User level scenarios

5.1.1 EGEE CLI level submission

This scenario focuses on existing EGEE users who are going to use the grid user interface machines they were using before. The validated application repository here is an aid to help EGEE users submitting applications which can be transferred to the desktop grid infrastructure if needed.

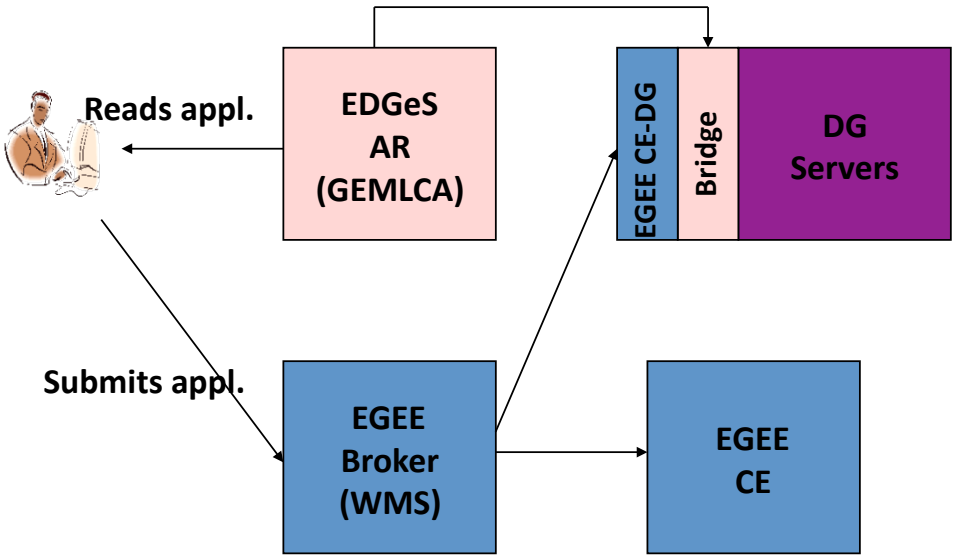

Figure 1: EGEE CLI level submission

1. The EGEE user lists applications available in AR. The list shows the available sites for the given application including EGEE and desktop grid nodes. This makes it possible for the user to prefer applications which are prepared for the desktop grid infrastructure.
2. User selects an application, and downloads the binaries compatible with the desktop grid.
3. User creates the JDL which refers to the binaries previously downloaded, and prepares the execution with the help of the application description gathered from a search request to the repository. The JDL can define a job collection in order to lower the burden on the WMS (gLite Workload Management System).
4. User submits the JDL to the WMS.
5. The WMS selects a regular CE (computing element), or a CE-DG (an EGEE computing element representing a desktop grid).
6. If the CE-DG receives the job then the job is passed to the EGEE-DG bridge. This leads us to the bridge level scenarios, for further details see section 5.3 on page 9.

5.1.2 WS-PGrade level submission

This scenario focuses on the existing PGrade portal users and the new EDGeS users who are encouraged to use the WS-PGrade portal. Since the WS-PGrade portal offers parametric study workflows the validated application repository should help the portal to decide to use EGEE or desktop grid infrastructure for the given job. The validated application repository should also help the workflow definition phase for the portal users by providing the list of applications which are available in the EDGeS infrastructure and offering the sites which can be used in conjunction with these applications.

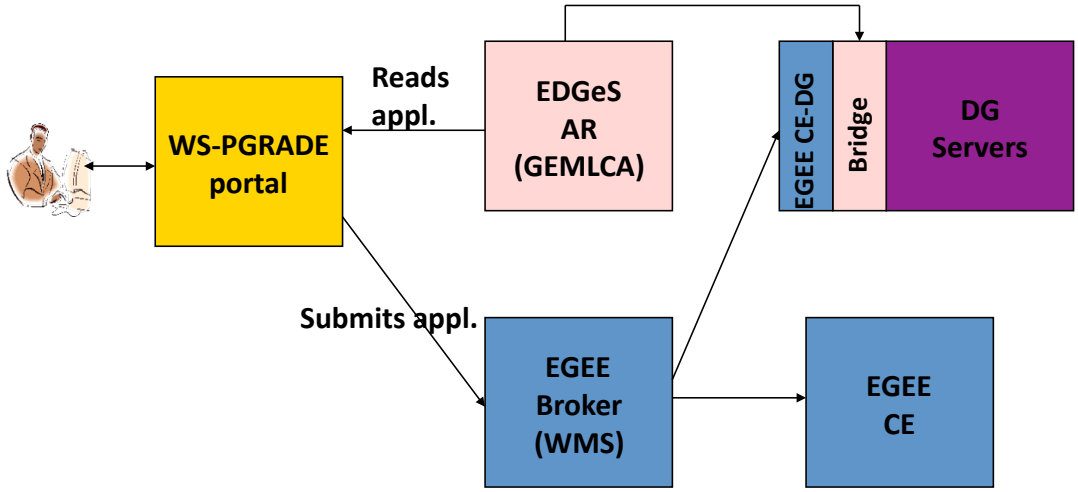

Figure 2: WS-PGrade level submission

1. Portal user lists applications available in AR. The list shows the available sites for the given application including EGEE and desktop grid nodes. Based on this information the portal can give hints to the user about how suitable is the application for parametric studies. This makes it possible for the user to prefer applications which are prepared for the desktop grid infrastructure.
2. The user selects an application.
3. The portal offers custom parametrisation options based on the description in the repository to simply create the JDL.
4. The user orders the portal to submit the job.
5. Based on the user selection the portal submits to the DG bridge directly or uses the WMS.

5.2 Site level scenarios

5.2.1 Application registration

This scenario deals with the registration of the validated applications in the repository. When the validated repository administrator receives a validated application it publishes the contents. The validated application can have multiple executables for the different grids and sites. Every executable is validated separately with its own description. Finally these descriptions are merged under one entry in the validated application repository.

Publication of the applications include the management of access rights. This means the application repository should restrict the use of its contents. Both the application descriptions and the files itself should be accessible on a controlled way.

Over the time applications will have newer versions, thus the repository administrator should revoke the old version and publish the newer ones instead. The repository might support deprecating entries in order to provide transitional periods between different versions of the application. This transitional period however have to be the shortest possible when serious flaws have been found in the older version. After the transitional period is over the executables and the related data should be purged from the repository.

5.2.2 Site registration

This scenario is initiated with a notification from the repository admin. The repository admin sends out notification messages every time a new application is introduced, an application changes or an application gets revoked. If a new application is added to the repository the desktop grid administrators decide whether they plan to support it locally. To support the application they investigate the contents of the repository, and adapt their sites accordingly. For example they download the necessary files for their specific desktop grid solution, and activate the application. Finally after the application is active on their desktop grid they register their sites to the application repository. This registration can involve the application repository administrator who can make sure the application description will hold the reference to the newly supported site. And also this registration would result that later on, their site can receive work units for the given application. In case a desktop grid administrator decided to support an application further notifications about that application should be followed.

As an example let's have a look at the BOINC registration procedure. BOINC servers also contain repositories which need to be synchronised with the validated application repository. In its repository each BOINC server maintains the applications which can be run by their clients. Therefore once the validated application repository administrator registered a new application in its repository, either the repository or the administrator must notify the desktop grid server admins (e.g. by e-mail). Then the desktop grid server admins download the new application from the repository which should hold all data necessary to register register in the desktop grid's repository. However this registration procedure is not mandatory (based on best effort), thus registration can be done with substantial delay or even the desktop grid admin can reject them because the type of application does not fit in the desktop grid's application portfolio. Therefore the contents of the desktop grid repositories are not exactly the same as the contents of validated application repository (only partial synchronisation is done).

Desktop grid admins, who trust the validation process, might decide to automate the application deployment process on their sites by running a program that deploys the application based on the data in the application repository.

5.3 Bridge level scenarios

5.3.1 Sending EGEE jobs to DGs

The repository has to support the bridge in two tasks. First the bridge has to be sure the submission received from the EGEE is a valid desktop grid application, and it should also offer

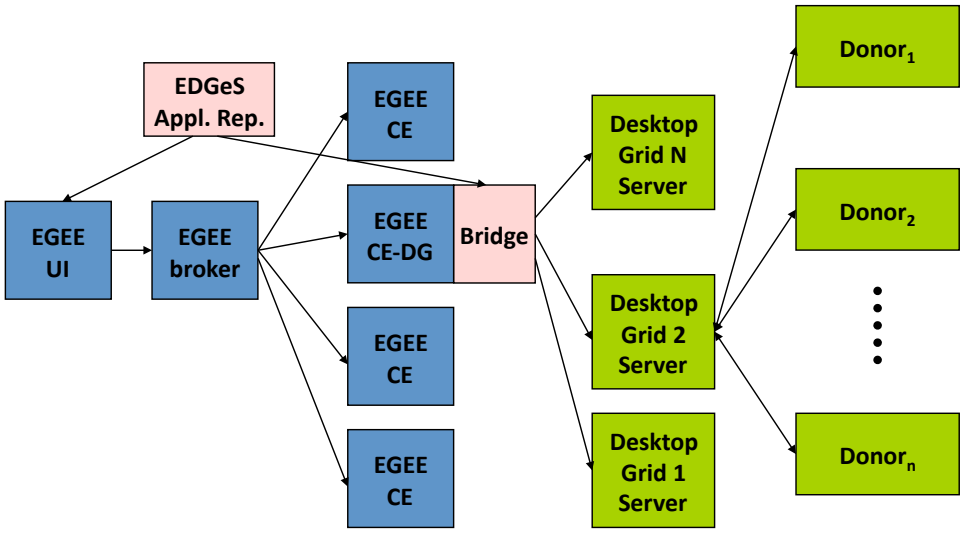

Figure 3: Bridge level scenarios

the list of desktop grid sites the application supports. The list of desktop grid sites should be filtered for the bridge depending on what kind of desktop grid it supports.

We assume that every EGEE virtual organisation that would like to use the EDGeS infrastructure should deploy a special computing element called as CE-DG which comes with the deployment of the EGEE to desktop grid bridge service. This bridge should be connected to a set of desktop grids (according an agreement between the virtual organisation and the desktop grid owners). This agreement does not mean that the desktop grid will support every application from the validated application repository. It means that the desktop grid supports some of the applications defined in the repository.

The bridge has the capability to group consecutive jobs, as an example this could help to provide better parametric study support. The bridge service should check which connected desktop grids support the EGEE job (which might define a job collection). The check is performed by reading and analysing the desktop grid registrations in the validated application repository. Then selects one or more desktop grids (a scheduling/brokering algorithm is needed here). When a single desktop grid is selected the entire set of previously grouped jobs are passed together, otherwise the grouping should be split to fit the capabilities of the selected desktop grids.

6 Application repository requirements

6.1 Inter-grid mapping

The application repository is crucial for the promised Service grid - Desktop grid interoperability because it holds information that enables bridges to transform tasks between grids of different concepts. Interoperability fails if info is not available or cannot be retrieved in the required form by the involved parties. One recurring misconception was “submitting jobs to Desktop Grids” which depending on the definition of terms may be another source of misunderstanding, because “Job” and “submit” are Service Grid concepts that most desktop grids do not readily support. For interoperability converting between concepts is also needed, this conversion is dependent on data in the repository.

A lot of SG concepts are the heritage of traditional local resource management systems, automating the way people used to run their executables by hand. Tasks are either thought of as jobs - “run this program as follows” - executable, parameters, resource requirements, etc. - jobs can only be run on appropriate resources. Alternatively tasks can be thought of as service invocations - “transform some data using this pre-installed service” - services can only be invoked after they are installed - services are usually not automatically deployed, only instantiated. EGEE follows the job based concept.

Desktop grids are based on very different concepts coming from the need to be able to use any available resources owned by non-experts to perform tasks (this is a major advantage of DGs that should be retained). For example the BOINC database of each project includes a set of platforms. A platform is a compilation target (which results in separate executables for each platform). An application usually includes several programs (different versions and different platforms) and a set of work units and results. A particular version, compiled for a particular platform, is called an application version. An application version can consist of multiple files and must be deployed before use. A work unit describes a computation to be performed by an application. The deployed master part of an application creates work units. A result describes an instance of a computation (either not started, in progress, or completed). A project does not explicitly create results; rather, BOINC creates them automatically for work units, based on the redundancy parameters of the work unit.

The validated application repository should help converting between security models described in table 1.

6.2 Interfaces, operations

As a basic requirement the repository must interoperate with different entities (portals, bridges, people uploading application data and managing rights) thus, must support access with different protocols and programming languages or be easily extensible for this. The repository should be securely accessible and capable to restrict access of its contents and should provide information about who is responsible for a given repository entry. In case of security problems or bugs this should enable to find who is responsible for publication.

Repository administering interfaces should be restricted and must not be used by anybody else but the EDGeS validated application repository administrators. Modification of the repository contents should not be allowed without revalidation of the modified entries.

Here is the minimal set of operations to be supported:

Create, read, update and delete (CRUD) applications in the repository

Service Grid	Desktop grid
Uses X509 public key infrastructure to authenticate users, resources and services and VOMS to authorise them	Thoroughly tested applications on all platforms and with all input data scenarios before promoting them to production status
User separation on resources and administered resources	Code signing with a key stored on a network-isolated machine to prevent malicious executable distribution (applications must be deployed by hand)
	Result validation and redundant computing to probabilistically detect result and credit falsification and result alteration due to differing platforms

Table 1: Grid security concepts compared

Retrieve list of application and additional data for the portal, and the EGEE cli users

Search applications by some criteria. (e.g. executable hash, version information, sites or grids supported)

Retrieve files from the repository when needed. Including executables and other data files like libraries or in case of BOINC signature files. This operation will be used by the EGEE to desktop grid bridges.

Determine application presence to find out whether an application is registered. This is similar to the search applications operation.

Determine ownership for the entries of the repository. Should give the list of people who contributed to the application. These people are responsible in case of security problems and errors.

Set and manage access rights on repository entries.

6.3 Contents to be stored

The repository must be able to store any kind of data without imposing format restrictions as we don't know all the data now and to allow extensibility (data are just like files or maybe like DB records for metadata that must be searched). Must support data organisation: connecting related data.

In order to support the user search function the repository application record should contain a field or fields describing the functionality of the application and its parameters.

In order to support the WS-PGrade portal, the repository should also contain necessary information about the supported grids and sites by each application.

Each application is described by an application record in the repository. This application record should hold generic application details and also grid specific data. The grid specific data should be enough in EGEE to submit a job. In practice this means all data which can be defined in a JDL should be definable. In case of desktop grids grid data is more extensive since it is split

into two parts. First it has to define data which is enough to create a work unit. Then it should also have enough information to register an application in the desktop grid's own repository if necessary. In the next sections we discuss this data in more detail.

6.3.1 BOINC

All data needed for registration should be in a single Zip file. Deploying the application consists of two steps: registering the *client application(s)* in the BOINC database, and running the *master daemon*. All client applications should be compiled for every platform you need, and installed under the project's `apps` directory. The master application must have access to the BOINC project's `config.xml` and be able to create files and directories under the project's download directory, and it must be able to access files under the project's upload directory. Besides the master and client applications, BOINC also requires supplying a *validator*[1] for the application. If you are not using redundancy to check results (e.g. because you trust all resources in a local desktop grid) then you may use the validator called `sample_trivial_validator` that comes with BOINC. This validator accepts everything without comparing results. If the results from an application are such that they can be bitwise compared then another default validator called `sample_bitwise_validator` may be used. Otherwise an application specific validator must be developed and supplied to BOINC.

As a preparation step for registration create a subdirectory for each application in your project's apps directory. Put new application files there. Create a directory with the same name as the main program (as *NAME_VERSION_PLATFORM[__PLAN-CLASS][.ext]*) and put the files in that directory. Example directory layout:

```
apps/ap/ap_7.17_windows_intelx86.exe/ap_7.17_windows_intelx86.exe
apps/ap/ap_7.17_windows_intelx86.exe/some_required_file.dat
apps/ap/ap_7.17_windows_intelx86.exe/graphics_application.exe
```

If a file of the form FILENAME.sig is found, its contents will be used as a digital signature for the corresponding file. All file entries should be available from the validated application repository.

Registration of boincified executables are done with the following command, see table 2 for details:

```
boinc_appmgr --add --client client.xml
```

This command places the executables where the BOINC expects them, then registers the client executables in the project's description file and in the BOINC database. Later on it installs the master application in a new directory then modifies the DC-API configuration to fit the master's needs, and configures the validator. Finally it makes sure the master application is automatically started or stopped when the BOINC do so. All *client.xml* should be available from the description of the repository entries representing the application.

The generic master also needs to be configured to produce work units for the client defining the application, its inputs, arguments and outputs. Then at least the Redundancy, MaxFPOps, MaxMemUsage, MaxDiskUsage, MaxOutputSize, DelayBound, EstimatedFPOps values should be defined to generate useful work units and results from the templates.

[1] See: http://boinc.berkeley.edu/trac/wiki/ValidationIntro

client.xml	
	name
	user_friendly_name
	version
platform	name
	binary
	binary signature
	library
	library signature

Table 2: Contents of the client.xml

6.3.2 XtremWeb

In case of XtremWeb the repository should present the following data:

- Application name
- Application owner
- Application access rights (Who can access what?)
- Application execution rights (Is it fully public or restricted in any manner?)
- Application binaries (where and how to access them, this should be a URI) for different platforms (win32, linux, mac os x. . .) and architectures (ix86, amd64, ppc . . .)

7 Comparison of the existing solutions

This analysis focuses on several important features of each of the considered repository technologies (see table 3).

Are these software technologies available/downloadable or are they existing only at the conceptual level? It would be better to reuse a piece of software if it is available and suits the needs of the workpackage, rather than implementing one from scratch.

What security infrastructures are supported? Grid had established a security infrastructure based on PKI concepts of authentication which was further enhanced with authorisation modules such as VOMS, SAML, Shibboleth, etc. The basic set of security modules and concepts found on Grid are grouped together under the name of GSI (Grid Security Infrastructure). Therefore, it is mandatory to know whether the software in discussion implement the GSI concepts or not.

Do these application storages support CRUD operations? The nature of the validated application repository requires the ability to browse through the content of these application containers. Users should be able to see the content of the repository and decide which application is suitable for them.

Do they employ search functions? Required to support the minimal operation set

Does the software in discussion support user-defined meta-model as well as user- defined meta-data? Both, the repository framework and registry framework should be able to accommodate particular, user-defined, meta-models. This would let the repository to be extended towards the support of the different grids, by creating new meta-models based on the info defined in section 6.3.

What hosting environment do they need? The repository should be as lightweight as possible while it should still provide the required features discussed before.

What kind of interfaces are offered with the repository? The repository should support a portable protocol, which is usable from the CLI the Bridges and also from the WS-PGrade portal.

Name	Description	Avail	Secure	CRUD	Search	Model	Hosting	Interf
Oracle 10g		Closed Source	SAML, GSI, WS-Sec	OK	OK	WS Only	WS Container	SOAP
IBM Web-sphere		Closed Source	SAML, XACML, WS-Sec, GSI	OK	OK	WS Only	WS Container	SOAP
ACS	OGF standard	Open Source simple imple-mentations	SAML, WS-Sec, GSI	OK	OK	OK	N/A	WSRF + others
WHCRP	WuHan Compo-nent Repository	–	–	OK	OK	OK	–	–
GEMLCA	University of Westminster	Open Source	WS-Sec, GSI	OK	Partial	Pluggable	WSRF Container	WSRF, JSR168, CLI, GridFTP
NGS JSDL		Closed Source	GSI	OK	OK	–	Portlet container	JSR168
ePrints		OK	–	OK	OK	–	HTTP server	HTTP
Gremoires		OK	GSI	OK	OK	OK	WS Container	SOAP
WenGReiC		Concept	GSI	OK	–	OK	–	–
OMII AHE	Application Host-ing Environment	OK	GSI	OK	Partial	–	WSRF-LITE	WSRF, GridFTP

Table 3: Comparison of the different repository technologies

As it can be seen from the table 3 the GEMLCA offers all necessary functionality. Furthermore it is developed and maintained by one of the project partners, and the WS-PGrade portal integration is already available. Thus it is chosen as the validated application repository. The necessary extensions and use as an EDGeS validated application repository is discussed in the later sections.

8 GEMLCA as the application repository

8.1 Introduction to GEMLCA

The actual version of the GEMLCA (Grid Execution Management for Legacy Code Applications) before the development was 3.2. This version included the basic repository functionality. Applications in the GEMLCA concept are called Legacy Codes (LC). The repository functionality is exhausted in the following capabilities:

CRUD operations for repository entry management.

Retrieve list of LCs with their human readable description or their id.

Searching with WSRF resource query based search for the LCs' id, status (public, private, system) and the executor sites supported by each LC.

Retrieve files with GridFTP from a non-hierarchical virtual filesystem. (Files related to a single LC should reside in the same directory.)

Retrieve application data in GEMLCA's proprietary format. Although application data is stored in an XML file this file cannot be accessed directly. Read operation parses the file to produce a list of XPath entries which identify single nodes in the XML DOM tree, and their current textual values. Examples:

```
/LCEnvironment/description,This is our very first LC
/LCEnvironment/parameter[@order='1']/value,Hello World
```

Determine application ownership based on the certificate subject the user provides.

Submit LC is one of the main functionality of GEMLCA however it is not needed in case of EDGeS validated application repository.

The GEMLCA Services can be accessed through various interfaces. First of all the GEMLCA team offers a Java based Client API. This API is clearly separating the concept of administration the concept of look-up and the concept of execution — ClientGLCAdmin, CliengGLCList and ClientGLCProcess classes. The team also offers the GEMLCA-PGrade portal as a user friendly interface to the GEMLCA services, with the portal the GEMLCA-Administration portlet is used for dealing with the repository interface meanwhile the Workflow editor and the script layer deals with the other two main functionalities. Then the team also offers a simple command line client. This client is delivered within the same jar set where the API was coming from, and offers the same functionality as the client API. It is recommended to use the GEMLCA CLI for building multi purpose clients. And finally the most unsupported way is to build your own clients from the GEMLCA WSDLs.

GEMLCA has a concept of backend plugin which defines the way how applications should be described, and also defines the way the execution starts on the selected site. GEMLCA plugins currently support service grids like GT2, GT4, gLite. Each plugin can require a different set of description to be provided for the application description. When a new LC is defined the GEMLCA checks what plugins it is using to define the LC and the union of all the descriptions required by those plugins will be asked from the user.

Security in GEMLCA is based on three factors. First the access to the GEMLCA service is controlled by a gridmap file. Secondly the access to the GEMLCA repository contents is

controlled by the OS — Unix file permissions. Thus during the installation of the EDGeS Validated application repository, the system administrator needs to adjust the rights for the GEMCLA repository on the following way:

1. EDGeSAR repository write permissions are restricted by regular Unix filesystem permissions
2. EDGeSAR admin group has write access on current LCIDs
3. EDGeSAR admin has right to create new LCID

Finally as the third security factor the GEMLCA defines two roles for each LC: the regular user and the owner. In GEMLCA only the owner has modification rights on a LC meanwhile the users only has the right for executions – if the LC is not private – for more details see section 8.2 on page 20.

GEMLCA has two WSRF services (GEMLCAFactoryService, GEMLCASErvice). One is singleton and deals with the creation of LC descriptions and the listing and searching of them. The other one WSRF resource is per LC description instance and represents the actual execution state when submitted, otherwise enables the modification and download operations. Since second resources' state (GEMLCAService resource) is not relevant for the repository functionality it is not discussed.

8.2 GEMLCAFactoryService resource

This resource is the singleton resource offered for MDS4 queries and searches:

```
<xsd:element name="legacycode">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="executorSite" minOccurs="0"
                                    maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="site" type="xsd:string"/>
       <xsd:element name="averageTimeSec" type="xsd:int"/>
       <xsd:element name="mdsState" minOccurs="0"
                                    maxOccurs="unbounded">
        <xsd:complexType>
         <xsd:attribute ref="common:state"/>
         <xsd:attribute name="number" type="xsd:int"/>
        </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="status" type="xsd:string"/>
 </xsd:complexType>
```

```
</xsd:element>
<xsd:element name="legacycodes">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="tns:legacycode" minOccurs="0"
                                     maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
<xsd:element name="GemlcaFactoryServiceResourceProperties">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="tns:legacycodes"/>
   <xsd:element ref="common:location"/>
   <xsd:element ref="metric:ServiceMetaDataInfo"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

List of notable elements or nodes:

legacycode is the element representing a single application.

name is the attribute name of the legacycode. Effective filtering can be done with this.

status is the visibility of the LC. There are three different statuses used in GEMLCA currently (common namespace defines more, but they are not used):

Public status shows that the LC is executable by everyone. You should make sure you never publish your applications (which are called Legacy codes or LCs in GEMLCA terms), private applications are just for you and can be used for testing purposes on the same way as using it publicly. The only difference that the application is not accessible by anyone else.

Private status means that only the owner of the Legacy code can execute and modify it. The ownership information is determined by the GEMLCA based on the proxy used to invoke the GEMLCA functions. The subject line is compared to the one stored in the repository within the application description.

System status is the GEMLCA's own. The LCs presented with system status are used heavily during GEMLCA's work.

executorSite holds the execution statistics for each site, this info is prepared for brokering. This is not relevant to the repository functionality so it is not discussed further.

8.3 The basic LC description

Top-level LCID Data Specification:

DESCRIPTION : Additional job description in human readable form. This description's primary aim is to support users to find applications by browsing the catalogues' descriptions.

ORDER : A number to specify the ordering of LC parameters in the list of runtime arguments. The order is used only in case of command line parameters.

ID : The global identifier of the job. Whenever you refer to it you use this id.

MAXIMUMPARALLELISM : Maximum number of parallel jobs of a process. From the P-Grade portal this value can be anything more than 1. The GEMLCA will enforce the maximum parallel executions per process. (A way to avoid some application licensing issues)

BackendSpecificData level LCID Specification:

COUNT : number of processes. Should be 1 unless submitting an MPI or multi job.

OUTPUT : name of the standard output file.

ERROR : name of the standard error output file.

JOBTYPE : The type of the job: single, multiple or MPI

maxWallTime : The maximum wall time for jobs (optional). After this many seconds pass the execution will not continue further.

Site definition:

JobManager : list of job managers supported by GEMLCA, from which select one.

Site : execution site(s) where the LC's executable can be executed

In GEMLCA concepts the generic application description is called parameter. This includes input output files, environment variables, or command line arguments also. The definition of the Parameters are here:

REGEXP : An optional field providing a Java regular expression to validate the LC's parameter value before submission. Client side support is required. GEMLCA does not check it on the server side whether this is really the proper value you have supplied.

COMMANDLINE : if it is ticked, the value of this parameter is going to appear on the command line argument list ordered by "order".

NAME : name of the argument as it appears in the command line (e.g. -r)

VALUE : default value of textual argument

FRIENDLYNAME : parameter's friendly name, which helps the user to understand what value does it have to specify when it starts to execute.

MANDATORY : If a LC parameter is mandatory, then prior submission the user should provide a different parameter value than the default one. If the parameter is not mandatory the user don't need to change the parameter value from the default one.

FIXED : parameters with default values that cannot be changed. Must not be used in conjunction with Mandatory field.

FILE : If true the value described by this parameter is a filename.

INPUT : if true the value described here is going to be used as an input file. Otherwise it will present the file as an output.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace=
  "http://uk.ac.wmin.cpc.gemlca/schema/legacyCodeConfig"
 xmlns:tns="http://uk.ac.wmin.cpc.gemlca/schema/legacyCodeConfig"
 xmlns:gp="http://uk.ac.wmin.cpc.gemlca/schema/generalParameters"
 elementFormDefault="qualified">

 <xsd:simpleType name="MinLenghtedString">
  <xsd:restriction base="xsd:string">
   <xsd:minLength value="1"/>
  </xsd:restriction>
 </xsd:simpleType>
 <xsd:element name="LCEnvironment">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="description" type="xsd:string"/>
    <xsd:element name="parameter" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="value" type="xsd:string"/>
       <xsd:element name="friendlyName" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="order" type="xsd:int" use="required"/>
      <xsd:attribute name="mandatory" type="xsd:boolean"
                     use="required"/>
      <xsd:attribute name="fixed" type="xsd:boolean" use="required"/>
      <xsd:attribute name="file" type="xsd:boolean" use="required"/>
      <xsd:attribute name="input" type="xsd:boolean" use="required"/>
      <xsd:anyAttribute namespace="##any" processContents="lax"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="backendSpecificData" minOccurs="1"
                 maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="siteInfo" minOccurs="1"
                    maxOccurs="unbounded">
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="site" type="xsd:string" minOccurs="0"
                       maxOccurs="unbounded"/>
          <xsd:element name="executable">
           <xsd:complexType>
            <xsd:simpleContent>
             <xsd:extension base="tns:MinLenghtedString">
              <xsd:attribute name="stage" type="xsd:boolean"/>
             </xsd:extension>
            </xsd:simpleContent>
           </xsd:complexType>
          </xsd:element>
          <xsd:element name="paramPrefix" type="xsd:string"
                       nillable="true"/>
```

```
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:anyAttribute namespace="##any" processContents="lax"/>
       </xsd:complexType>
      </xsd:element>
     </xsd:sequence>
     <xsd:attribute name="backendId" type="xsd:string" />
     <xsd:anyAttribute namespace="##any" processContents="lax"/>
    </xsd:complexType>
    <xsd:unique name="siteInfoId">
     <xsd:selector xpath="tns:siteInfo"/>
     <xsd:field xpath="@id"/>
    </xsd:unique>
   </xsd:element>
   <xsd:element name="authorizationInfo">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="owner" type="xsd:string" nillable="true"/>
      <xsd:element name="email" type="xsd:string" nillable="true"/>
      <xsd:any minOccurs="0"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="status">
   <xsd:simpleType>
    <xsd:restriction base="xsd:string">
     <xsd:enumeration value="offline"/>
     <xsd:enumeration value="public"/>
     <xsd:enumeration value="private"/>
     <xsd:enumeration value="submit_pre"/>
     <xsd:enumeration value="submit_ready"/>
     <xsd:enumeration value="submitted"/>
     <xsd:enumeration value="system"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="maximumParallelism" type="xsd:int"/>
 </xsd:complexType>
 <xsd:unique name="parameter">
  <xsd:selector xpath="tns:parameter"/>
  <xsd:field xpath="@order"/>
 </xsd:unique>
</xsd:element>
</xsd:schema>
```

There is a close relation between what is in the application description and what can be downloaded from the repository URLs shown in table 4. To determine whether a nonfixed file is default you have to get the parameters of the LC under investigation. The last returned item for each parameter is going to be whether the argument is in repository or not. The download url is constructed by concatenating the lcStorageURL of a given LC and the current value of the parameter. The lcStorageURL is always a GridFTP url in the current implementation which points to a location reserved for LC storage on the GEMLCA site.

In case of executables (an application can have as many executables as many siteinfo elements it has) the situation is much easier. If the stage attribute is true on the executable, then it

Fixed	File	Input	Default	Downloadable from
true	true	true	?	$lcStorageURL/$Value
false	true	true	NonDefault	–
false	true	true	Default	$lcStorageURL/$Value
?[a]	false	?	?	–
?	?	false	?	–

[a]indefinite state

Table 4: Downloadable entries from the repository

can be downloaded from the following URL: `$lcStorageURL/$executable`

8.4 Necessary extensions

8.4.1 Operations

As it can be seen from the list of operations provided and the list of operations required, the GEMLCA 3.2 lacks the operation Determine application presence and searching for application versions.

Determine application presence is solved with the help of already existing repository search operation, and the newly created repository file hashing capabilities just added to GEMLCA:

1. Generic bridge receives an executable for EGEE submission
2. Generates the MD5 hash of the received executable
3. Generic bridge queries the list of LCs from the EDGeS validated repository (previously existing feature)
4. Generic bridge *queries* EDGeS validated repository for each registered LC's *executable hashes*. This would result several executables per LC (at least one per each backend). For example:

```
$gemlcacli -getFileHashes -proxy GENBrProxy \
https://edgesar.cpc.wmin.ac.uk:8443/wsrf/\
services/uk/ac/wmin/cpc/gemlca/frontend \
exampleBoincifiedLCID
```

5. Generic bridge compares the received hashes with the MD5 generated at the first step Only the hashes for EDGeSBroker backend are checked
6. Once a match is found the bridge collects the BOINC sites supported by the matching LC

The hashing capabilities of GEMLCA now let the users query all file hashes for a given LC.

Searching for application version is solved on two ways. First of all a new operation was introduced on the GEMLCAFactoryService which is capable of retrieving LC descriptions newer than a given time: `listLCIDsFrom(unixdate)` Example GEMLCA CLI command:

```
$gemlcacli -listFrom -proxy $X509_USER_PROXY \
https://edgesar.cpc.wmin.ac.uk:8443/\
wsrf/services/uk/ac/wmin/cpc/gemlca/frontend \
'06/13/2008 08:00 AM'
```

The second solution provides more flexible searching facilities because it offers the date in the GEMLCAFactoryService's resource. This means standard WSRF Query operations can be called for filtering, and further client side (e.g.XML processing) operations can also be applied (e.g. XSLT) on the resource document before using its contents. The new resource property portion is called "since" and it is introduced as an attribute to the legacycode resource property. An example for the standard query mechanism (using the embedded client in Globus Toolkit 4):

```
$wsrf-query -z none -s https://edgesar.\
cpc.wmin.ac.uk:8443/wsrf/services/uk/ac/wmin/cpc\
/gemlca/frontend/GemlcaFactoryService \
| grep 2008-06-13
```

Section 5.2.1 requires that modification should not be allowed even for the repository administrators. This restriction is not possible to make yet in GEMLCA. But this restriction can be enforced by simply asking the administrators not to do so since there will be not too many administrators.

8.4.2 Site registration

Whenever a new application is added to the repository the EDGeS validated application repository administrator sends out an email about its details to the mailing list of the EDGeS dekstop grid administrators. Receiving the mail lets the admins decide whether they plan to install the application or not.

If they plan to, then the GEMLCA Admin portlet should support downloading the repository contents. Through this interface the BOINC admins should be able to download the application bundle. This interface is ready for use, however some admins might prefer using GridFTP, then they should have a look on table 4 for more details.

BOINC admins should be able to register their sites after they deployed the application locally: Application description's site section should be modifiable by the BOINC admins (other sections are exclusively written by the EDGeS AR admins). Right management in interfaces were introduced in GEMCLAService: listAppendableDescriptionProperties, listModifiableDescriptionProperties. These functions are used to determine whether a specific entry in the LC description is modifiable by the user who makes the request. In case of the BOINC admin this would mean the admin queries whether it can append to a specific LC's siteinfo element. If the BOINC admin is listed in a special grid-mapfile, then the GEMLCA will let the appendDescriptionProperty function to succeed.

```
$echo /LCEnvironment/backendSpecificData[@back\
endId='boinc']/siteInfo[@id='1']/site,szdg.lp\
ds.sztaki.hu > newboincsite
$gemlcacli -appendDescProps -proxy $X509_USER_PROXY\
 https://edgesar.cpc.wmin.ac.uk:8443/wsrf/services/\
```

```
uk/ac/wmin/cpc/gemlca/frontend exampleBoincifiedLCID\
 newboincsite
```

The two right listing functions are used by the GEMLCA admin portlet to prevent unauthorised operations before they even get to the GEMCLA service. Thus the service only lets change those description parts which can be modified or appended by the given admin portlet user. With this extension a new role is introduced to GEMLCA and now the list is threesome: Owner, BOINC Admin, User. The BOINC admin has the privilege to append sites to the LC, but otherwise behaves as a regular user, not having ownership on the LC.

8.4.3 Repository contents

The repository can store any kind of files. The executable data which should be used by the xtremweb sites should be held in one of the siteinfo elements of the xtremweb backend.

The GEMLCA is used as a repository in this case the BOINC and XtremWeb plugins. Thus the plugin is not going to execute anything, it is just going to offer the descriptions of the applications on a BOINC or xtremweb manner. This description should be proposed by the BOINC and xtremweb experts later on. The executables and other important files for execution should be stored as a single package under the corresponding backends siteinfo element. If there are more executable setups available then they can be packaged separately for every new desktop grid site.

As for security only the EDGeSAR VO members can access the files in the repo, and even they should access them with their proxy certificates. However it has to be stated that currently anybody who has a proxy extended by the EDGeSAR VOMS will be able to download everything from the EDGeS validated application repository. Only upload operations are restricted to the repository admins. Since all EDGeSAR VO members can access the application binaries and default input files stored in the repository, the users will have the possibility to submit EGEE jobs differently than the GEMLCA or the generic bridge would do it for them. Different submission in our case would mean a handmade JDL or a replaced input file.

8.5 Further information

CLI Guide, APIs and WSDLs can be found at the EDGeS SVN:

```
edges/sa1/devel/ValidatedAR
```

The GEMLCA user community is just getting diverted to the official globus mailing lists thus please subscribe and discuss the further questions here:

```
http://dev.globus.org/wiki/Incubator/GEMLCA
```

8.6 Conclusion

GEMLCA is now able to handle the basic requirements of the EDGeS users. Its extensions towards versioning and access control prepares the further steps ahead in order to support the EDGeS infrastructure better. There are ongoing discussions about the directions of future enhancements, here are some of the more important ones:

Automated DG admin notification. It is possible to send out automated emails by using the GT4's MDS4trigger service to monitor the GEMLCAServiceFactory's resource property changes and fire emailing events on changes. The GEMLCA team might also offer a script which can be ran on the BOINC site and which after it realises a new or a modified LC is available automatically downloads to the BOINC site, so later on only the registration task remains for the dg admin.

Dry runs. The GEMLCA should help creating the JDLs and selecting the necessary input files and executables for command line EGEE the users. This would mean the GEMLCA job execution should be split in two phases, the first being the job description generation and the latter is the job submission itself. If the execution is split, then it should be possible for the user to download the generated JDLs from the GEMLCA service.

Lightweight CLI client. The GEMLCA now offers Java only clients which are time consuming to load and these clients are usually don't do to much in one session. Therefore the clients spend their time mostly with loading the JVM. This can be avoided by using a native CLI client, and globus offers C bindings to their WSRF core also. Thus a C client might lower the burden on the user's side. Meanwhile this client gets designed and ready there is a servlet based client which accepts localhost connections and can be used as the CLI client for most tasks. Using the servlet means the JVM is only loaded once. However this servlet can only be used for services like the generic bridge.

Non bundled file addition. Currently if the application needs more files then they can be included on two ways. First a archive bundle can be created and put in the location of the real exe. This bundle can hold all the executables needed for the application. However the bundling the files means they should be downloaded together, and if only a small portion of the bundle is updated then the dg admin has to figure out on its own what was the change in the bundle. Secondly the GEMLCA offers its fixed input files facility. With this facility the EDGeS validated application repository admin can upload multiple files to the repository without bundling them. However this means the GEMLCA will handle the file as input file and will transfer it on every submission it does. For example this would mean that even though an EGEE submission took place the BOINC executables are also transferred to the EGEE worker node.

A possible solution can be to make parameters conditionally available in the different plugins. The condition can be expressed by an extra element called *onlyinbackend* in the */LCEnvironment/parameter*. A parameter with this onlyinbackend element would be accessible from only the specified backend. Therefore it still can be defined as a fixed input without having to fear about unwanted transfer of the input file.

Better access control. Legacy codes in GEMLCA now only have two level of views. Public legacy codes can be seen by everybody and private legacy codes can only be seen by their owners. In the future the EDGeS validated application repository admin should be capable of defining access privileges between public and private. It would be the simplest way to use the VOMS attributes the users add to their proxies before they contact the repository. The admin just have to specify what extension it requires to access a given LC and the more fine grained access control is already available.

9 The desktopgrid VO

9.1 The role of the VO

EGEE is organised in Virtual Organisations (VOs). A VO contains entities that are grouped together around some task or project. A Virtual Organisation includes both services and people: resource brokers, Computing Elements (CE), Worker Nodes (WN), Storage Elements (SE), file catalogues and information system are on the service's side and users, administrators and people with other roles are on the people's side. People in a VO (or members of a VO) are allowed to use the VO services. EDGeS set off to create a VO to collect resources and people involved in connecting desktop grid systems and a service grid, EGEE.

In the current phase of EDGeS when the DG→EGEE direction is elaborated the VO is used to collect EGEE resources that are accepting jobs bridged from desktop grid systems. At this point the VO is not open for users, no user can register to the VO, only the bridges can submit jobs to the resources. Resources (mainly CEs) however, can be configured to accept this VO. So far the gLite resources of the EDGeS partners have been configured to support this VO but in the future we encourage more resources to join from EGEE or other sources such as the EELA-2 collaboration.

9.2 Relation to EGEE

The EDGeS desktopgrid VO is created independently by EDGeS but the registration of this VO as an External VO in EGEE is underway. The expected outcome of this process besides allowing EGEE resources outside of EDGeS to also join this VO is also to be able to use the operational support infrastructure already put in place by EGEE which helps in day to day operation of the VO and also providing help for user support later in the 2nd phase of EDGeS when users will also start to use the VO.

9.3 VO Operation

The VO is managed by SA1, the VOMS server hosting this VO is operated by MTA SZTAKI. The host name of the server is `voms.grid.edges-grid.eu`. The certificate of the server is the following:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 839 (0x347)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=HU, O=NIIF, OU=Certificate Authorities, CN=NIIF Root CA
        Validity
            Not Before: May 19 14:47:08 2008 GMT
            Not After : May 19 14:47:08 2009 GMT
        Subject: C=HU, O=NIIF CA, OU=GRID, OU=MTA SZTAKI,
                 CN=voms.grid.edges-grid.eu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
```

```
            Modulus (1024 bit):
                00:ad:37:27:41:98:fd:fd:ea:2d:8b:65:1f:3a:4b:
                cb:55:c9:40:2a:dc:6e:59:cf:56:af:6e:7b:b1:48:
                84:a5:47:8b:70:af:47:93:cb:a8:7d:82:5e:2b:50:
                72:a5:f3:4d:bf:44:3d:a0:9a:f8:61:60:cf:d4:15:
                8b:06:72:71:ff:09:7d:af:b4:ca:17:66:42:4f:22:
                6a:4b:9c:f1:1c:d4:2b:fb:df:af:ff:ca:2e:d2:99:
                67:79:02:21:ba:9b:f8:79:c4:06:65:68:75:70:fe:
                f9:86:65:1d:10:a0:96:56:26:f2:45:27:c9:e5:ac:
                e0:fb:78:4b:58:53:f1:da:6f
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            Netscape Cert Type:
                SSL Server
            X509v3 Subject Alternative Name:
                email:balaton@sztaki.hu
            X509v3 CRL Distribution Points:
                DirName:
                  /C=hu/O=NIIF/OU=Certificate Authorities/CN=NIIF Root CA
                reasons:<UNSUPPORTED>
                CRLissuer:<UNSUPPORTED>
                URI:http://www.ca.niif.hu/niif-ca-crl.crl
                reasons:<UNSUPPORTED>
                CRLissuer:<UNSUPPORTED>

            X509v3 Authority Key Identifier:
                keyid:8C:6E:21:E2:71:AF:A0:2A:A7:B0:
                      E4:FE:BC:7E:A3:FD:0F:A0:E3:88

            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.11914.1.1.5.1.3

            X509v3 Basic Constraints: critical
                CA:FALSE
    Signature Algorithm: sha1WithRSAEncryption
        6d:59:00:de:89:57:98:e7:1a:b8:54:ba:4a:21:41:15:3c:3d:
        ff:44:31:8d:15:17:87:47:9c:30:dd:76:2f:bb:e8:fb:a2:31:
        5f:b7:5d:26:92:96:53:33:68:02:93:1f:91:76:64:ef:e6:cd:
        4d:43:22:70:15:f3:a9:f9:2b:01:2e:77:86:ac:dd:51:d6:a6:
        59:9c:7a:89:09:2d:8f:b8:1e:d5:1d:85:57:fa:be:8e:e4:b8:
        58:a9:cf:c6:e4:f6:5a:e0:b6:8b:ec:73:be:e5:b9:39:2d:5c:
        bb:0d:9d:84:da:25:7e:69:c1:41:22:73:de:89:fc:ac:91:53:
        7b:ca:69:95:cc:58:cb:6b:a8:28:dd:af:d8:87:44:32:a6:30:
        b7:b6:e0:a0:fb:60:1f:1d:16:4d:95:21:c9:6b:10:46:8a:6a:
        a6:2b:fb:b0:17:fd:9b:f0:5d:ac:df:31:cc:17:69:e0:5c:fd:
```

```
        38:0f:78:90:ed:da:e6:9d:79:cc:fa:cf:96:33:2c:b1:97:2e:
        47:20:7b:35:bd:6e:e4:aa:03:27:7b:27:87:f1:2e:56:a8:ac:
        c1:39:ac:50:37:5e:de:e4:97:6c:89:1a:28:b5:79:bc:fe:03:
        ee:4f:ea:ee:84:e3:a5:bd:04:a8:a7:cf:65:cb:72:f8:38:a1:
        46:6d:c4:18
-----BEGIN CERTIFICATE-----
MIIEnTCCA4WgAwIBAgICA0cwDQYJKoZIhvcNAQEFBQAwVTELMAkGA1UEBhMCSFUx
DTALBgNVBAoTBE5JSUYxIDAeBgNVBAsTF0NlcnRpZmljYXRlIEF1dGhvcml0aWVz
MRUwEwYDVQQDEwxOSUlGIFJvb3QgQ0EwHhcNMDgwNTE5MTQ0NzA4WhcNMDkwNTE5
MTQ0NzA4WjBlMQswCQYDVQQGEwJIVTEQMA4GA1UEChMHTklJRiBDQTENMAsGA1UE
CxMER1JJRDETMBEGA1UECxMKTVRBIFNaVEFLSTEgMB4GA1UEAxMXdm9tcy5ncmlk
LmVkZ2VzLWdyaWQuZXUwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAK03J0GY
/f3qLYtlHzpLy1XJQCrcblnPVq9ue7FIhKVHi3CvR5PLqH2CXitQcqXzTb9EPaCa
+GFgz9QViwZycf8Jfa+0yhdmQk8iakuc8RzUK/vfr//KLtKZZ3kCIbqb+HnEBmVo
dXD++YZlHRCgllYm8kUnyeWs4Pt4S1hT8dpvAgMBAAGjggHpMIIB5TALBgNVHQ8E
BAMCBeAwEQYJYIZIAYb4QgEBBAQDAgZAMBwGA1UdEQQVMBOBEWJhbGF0b25Ac3p0
YWtpLmh1MIIBWQYDVR0fBIIBUDCCAUwwgbygW6BZpFcwVTELMAkGA1UEBhMCaHUx
DTALBgNVBAoTBE5JSUYxIDAeBgNVBAsTF0NlcnRpZmljYXRlIEF1dGhvcml0aWVz
MRUwEwYDVQQDEwxOSUlGIFJvb3QgQ0GBAgDeolmkVzBVMQswCQYDVQQGEwJodTEN
MAsGA1UEChMETklJRjEgMB4GA1UECxMXQ2VydGlmaWNhdGUgQXV0aG9yaXRpZXMx
FTATBgNVBAMTDE5JSUYgUm9vdCBDQTCBiqApoCeGJWh0dHA6Ly93d3cuY2Eubmlp
Zi5odS9uaWlmLWNhLWNybC5jcmyBAgDeolmkVzBVMQswCQYDVQQGEwJodTENMAsG
A1UEChMETklJRjEgMB4GA1UECxMXQ2VydGlmaWNhdGUgQXV0aG9yaXRpZXMxFTAT
BgNVBAMTDE5JSUYgUm9vdCBDQTAfBgNVHSMEGDAWgBSMbiHica+gKqew5P68fqP9
D6DjiDAZBgNVHSAEEjAQMA4GDCsGAQQB3QoBAQUBAzAMBgNVHRMBAf8EAjAAMA0G
CSqGSIb3DQEBBQUAA4IBAQBtWQDeiVeY5xq4VLpKIUEVPD3/RDGNFReHR5ww3XYv
u+j7ojFft10mkpZTM2gCkx+RdmTv5s1NQyJwFfOp+SsBLneGrN1R1qZZnHqJCS2P
uB7VHYVX+r6O5LhYqc/G5PZa4LaL7HO+5bk5LVy7DZ2E2iV+acFBInPeifyskVN7
ymmVzFjLa6go3a/Yh0QypjC3tuCg+2AfHRZNlSHJaxBGimqmK/uwF/2b8F2s3zHM
F2ngXP04D3iQ7drmnXnM+s+WMyyxly5HIHs1vW7kqgMneyeH8S5WqKzBOaxQN17e
5JdsiRootXm8/gPuT+ruhOOlvQSop89ly3L4OKFGbcQY
-----END CERTIFICATE-----
```

The VO naming follows the DNS like naming convention recommended by EGEE and is called `desktopgrid.vo.edges-grid.eu`.

9.4 Configuring resources to support the VO

To configure resources to accept the `desktopgrid.vo.edges-grid.eu` VO the following steps are necessary:

1. Configure the EDGeS repository and install the VOMS certificate:

   ```
   cd /etc/yum.repos.d/
   wget http://intraweb.edges-grid.eu/public/grid_deployment/edges.repo
   yum install edges-vomscerts
   ```

2. Add pool accounts and mappings to users.conf and groups.conf:

- users.conf:

```
27000:desktopgsgm:27000:desktopg:desktopgrid.vo.edges-grid.eu:sgm:
27001:desktopg001:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27002:desktopg002:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27003:desktopg003:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27004:desktopg004:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27005:desktopg005:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27006:desktopg006:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27007:desktopg007:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27008:desktopg008:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27009:desktopg009:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27010:desktopg010:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27011:desktopg011:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27012:desktopg012:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27013:desktopg013:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27014:desktopg014:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27015:desktopg015:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27016:desktopg016:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27017:desktopg017:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27018:desktopg018:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27019:desktopg019:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27020:desktopg020:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27021:desktopg021:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27022:desktopg022:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27023:desktopg023:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27024:desktopg024:27000:desktopg:desktopgrid.vo.edges-grid.eu::
27025:desktopg025:27000:desktopg:desktopgrid.vo.edges-grid.eu::
```

- groups.conf

```
"/VO=desktopgrid.vo.edges-grid.eu/
    GROUP=/desktopgrid.vo.edges-grid.eu"::::
"/VO=desktopgrid.vo.edges-grid.eu/
    GROUP=/desktopgrid.vo.edges-grid.eu/ROLE=SGMAdmin":::sgm:
```

3. Add VO configuration to vo.d named desktopgrid.vo.edges-grid.eu:

```
SW_DIR=$VO_SW_DIR/desktopgrid.vo.edges-grid.eu
DEFAULT_SE=$CLASSIC_HOST
STORAGE_DIR=$CLASSIC_STORAGE_DIR/desktopgrid.vo.edges-grid.eu
VOMS_SERVERS="vomss://voms.grid.edges-grid.eu:8443/
  voms/desktopgrid.vo.edges-grid.eu"
VOMSES="'desktopgrid.vo.edges-grid.eu voms.grid.edges-grid.eu 15000
  /C=HU/O=NIIF CA/OU=GRID/OU=MTA SZTAKI/CN=voms.grid.edges-grid.eu
  desktopgrid.vo.edges-grid.eu'"
```

Note: Due to a bug in yaim currently VOMS_CA_DN should not be specified.

4. Add or change the following lines in site-info.def as needed:

```
VOS="desktopgrid.vo.edges-grid.eu"
QUEUES="desktopg"
DESKTOPG_GROUP_ENABLE="desktopgrid.vo.edges-grid.eu"
GRID_TRUSTED_BROKERS="
'/C=HU/O=NIIF CA/OU=GRID/OU=MTA SZTAKI/CN=wms.grid.edges-grid.eu'
"
```

5. Run `yaim` to reconfigure your CE
6. If you're behind a firewall you should allow `193.224.187.128/25` to access your services.
7. Finally send the following information to the VO manager (reachable at `edges-vomsadm@mail.edges-grid.eu`):
   - SITE_NAME (as set in site-info.def, the site name you are publishing in BDII)
   - The URL of your site BDII (something like `ldap://host:port/`)
   - The IP addresses of your CE and all WNs (can be an address range)

9.5 Further information

The latest version of the updated instructions in the future can be found at:

`http://intraweb.edges-grid.eu/public/grid_deployment/`