Trust and Security for Next Generation Grids

D4.3 – VO Management System

Deliverable:	D4.3	**Version:**	1.0	**Status:**	Final	**Date:**	30/5/2009	**File:**	GridTrust-D4 3-V2.0a-reviewed
Workpackage:	WP4	**Task:**	T4.3	**Title:**	VO Management				
Leading Organisation:	INT	**Participant Organisations:**	INT, MOV						
Reviewers:	VUA, CETIC	**Dissemination Level:**	PU - Public						

Table of Contents

Document History

Date	Version	Description	Author	Partner
30/6/2008	1.0	First release (TRS design only)	G. Cortese, S. Riz B. Aziz, A. Arenas	Interplay Software STFC
30/6/2009	2.0	Final release (TRS and VBE/VO manager)	G. Cortese, S. Riz A. Sánchez	Interplay Software Moviquity

1. INTRODUCTION

1.1 Scope and goals

This document is the final output of GridTrust tasks T4.3 and T4.4, together with the software which is available on the Sourceforge project repository [5].
It reports on the design and implementation of the GridTrust Trust and Reputation (TR) service and VO Management services.
The Trust and Reputation Management system will produce tools for collecting, verifying and assessing evidence about service performance (including compliance with SLA and security policies) for the purpose of measuring the trustworthiness of a service provider. It includes tools for maintaining and disseminating reputation information as well as assessing the trustworthiness of a service by combining evidence about performance with reputation information and recommendations.
The design is based on the foundations of the model described in **Erreur ! Source du renvoi introuvable.**, which has also been developed as part of this task.

The VBE coordinates the process of VO creation and the registration in the VO of users and service providers. The VBE stores the required information in the PPM and the TRS and with the CA creates the required VO certificates.

The VO library makes the communication with the VBE and the management of VO certificates easier.

The GridTrust util library provides classes to manage certificates, private and public keys, proxy certificates, PKCS12 stores, submit jobs, etc.

Design of these components is also based on the internal document which describes the overall Gridtrust framework **Erreur ! Source du renvoi introuvable.**, and on the OGSI Specification **Erreur ! Source du renvoi introuvable.**. The implementation is based on Globus toolkit.

1.2 Structure of the document

This document provides documentation useful to those who plan to use the VO Management services and Trust and Reputation service provided by GridTrust.
Each section includes subsections briefly covering:

- Main concepts and functionalities
- Architecture/ Prototype Description
- Usage information – guidelines for developers willing to use the component

2. VBE/ VO MANAGEMENT SERVICES

2.1 Concepts and Functionalities

The VBE Manager coordinates the process of users and service providers registration, VO creation and the registration in the VO of users and service providers. The VBE Manager registers the users, services providers, VOs and VO users and services providers in the PPM and TRS and with the CA module creates the users, services providers and VO certificates.

2.2 Architecture and Prototype description

When the registration of a new user or service provided is requested the VBE Manager registers the new user or service provider in the PPM and the TRS and creates a VBE user or service provider certificates.

When the creation of a new VO is requested the VBE Manager checks the client identity in the VBE certificate, registers the new VO in the PPM and the TRS, creates a VO Owner Certificate with the CA module an returns this certificate to the VO creation requestor.

When the registration of a User or Service Provider in a VO is requested the VBE Manager checks the client identify in the VBE certificate and validity of the VO Owner certificate, registers the VO User or Service Provider in the PPM and the TRS, creates a VO User or VO Service Provider Certificate and returns this certificate to the VO Owner or sends it to the Service Provider.

The VO Manager Library wraps the VBE interface and makes the communication with the VBE and the management of VO certificates easier.

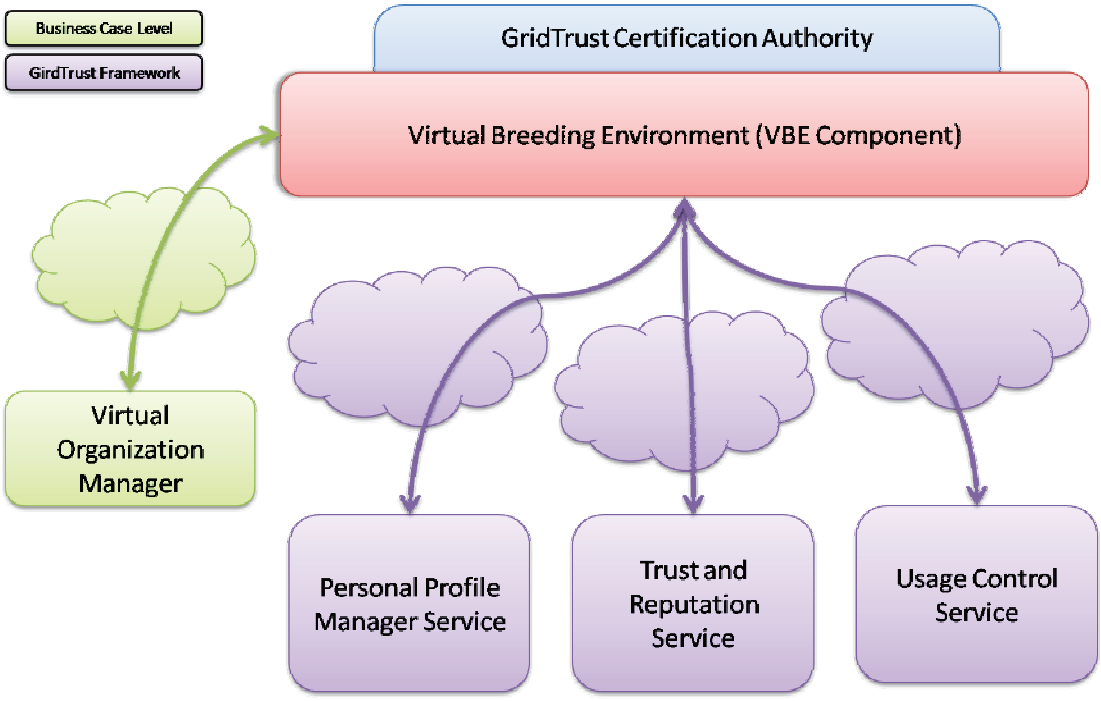

Figure 1 - PLEASE ADD A DESCRIPTION OF THE FIGURE HERE

2.2.1 *Certificate Authority*

The VBE Manager is the System Certificate Authority. The CA uses the library Bouncy Castle [12] to generate X509 certificates [6]. The CA uses the certificate and private key of a Globus SimpleCA [7] in PEM format for signing certificates.

The use of the certificate and private key of a Globus SimpleCA has the advantage of the easy integration of the VBEManager with the Globus GSI (Grid Security Infrastructure).
With a SimpleCA creation is created a "CA setup package". Trust in the VBEManager CA must be setup in all servers and clients with this package, file globus_simple_ca_xxxxxxxxxx_setup-0.19.tar.gz, and the following commands:

```
$GLOBUS_LOCATION/sbin/gpt-build globus_simple_ca_ xxxxxxxxxx _setup-0.19.tar.gz
$GLOBUS_LOCATION/sbin/gpt-postinstall
```

This is a standard process described in the Globus Quick Start Guide [8].

2.2.2 *Certificate types*

There are three standard types of certificates: VBE Certificates, VO Certificates and GridTrust services certificates. VBE Certificates identifies the users or service providers and VO certificates give permissions over VOs.

Each certificate type has a unique OID. These OIDs are defined in the java class GridTrustCertificate.

2.2.2.1 VBE Certificates

There are two types of VBE certificates:

- VBE User Certificate: The certificate that identifies the client as a VBE user. This certificate has the following information:
  - ID (user email)
  - Country
  - State
  - Locality
  - Organization
  - Organizational Unit
  - Common Name (full user name)
  - Email address

- VBE Service Provider certificate: The certificate that identifies the provider as a VBE service provider. This certificate has the following information:
  - ID (service provider host name)
  - Country
  - State
  - Locality
  - Organization
  - Organizational Unit

- Common name (service provider host name)
- Email address (service provider administrator email address)

2.2.2.2 *VO Certificates*

There are three types of VO certificates:

- VO Owner Certificate: This is the certificate that gives control over a VO It has the following extensions:
  - ID (VO ID)
  - Owner Id (User or Service provider ID)

- VO User Certificate: This is the certificate given to VO users. It has the following extensions:
  - ID
  - VO ID
  - Owner Id (User ID)

- VO Service Provider Certificate: This is the certificate given to service providers. It has the following extensions:
  - ID
  - VO ID
  - Owner Id (Service Provider ID)

2.2.2.3 *GridTrust Services Certificates*

These certificates have the same information than VBE Service Providers Certificates and are used by the GridTrust services like PPM, TRS, etc.

2.2.3 *Security System*

The security system uses Globus Security Infrastructure (GSI) [9]. GSI is configured with configuration files named Security Descriptors [10]. GSI has two communications modes: "GSI Secure Conversation" and "GSI Transport". GSI Secure Conversation uses WS-Security and only encrypts the SOAP message body. GSI Transport uses HTTPS and encrypts all the communication. The mode can be changed in the configuration file without modifying the service.

2.2.3.1 *Client setup*

Globus libraries and utils implementing GSI and WSRF must be installed to connect to Globus web services with GSI security. The simplest way is installing full Globus Toolkit. Libraries in the folder $GLOBUS_LOCATION/lib and the file $GLOBUS_LOCATION/client-config.wsdd must be in the Java classpath

2.2.3.2 *Client GSI Anonymous Communication*

Clients can execute methods like VBEManager.registerUser and VBEManager.registerServiceProvider before owning a certificate by using an encrypted and "anonymous" communication with the following parameters and security descriptor:

```
String CLIENT_DESC = /home/user/client-security-config-anonymous.xml);
((Stub))vbe)._setProperty(Constants.CLIENT_DESCRIPTOR_FILE, CLIENT_DESC);
```

The content of client-security-config-anonymous.xml is:

```
<securityConfig xmlns="http://www.globus.org">
    <GSISecureConversation>
        <privacy/>
        <anonymous/>
    </GSISecureConversation>
</securityConfig>
```

With that security descriptor we make an encrypted and "anonymous" communication without a certificate, register and get a X.509 certificate (User Certificate or Service Provider Certificate).

2.2.3.3 *Client GSI Authenticated Communication*

When the user or service provider client is registered in the VBE and has a certificate and a private key it can configure the security descriptor with the path to the certificate and private key.

```
<securityConfig xmlns="http://www.globus.org">
        <credential>
                <key-file value="/home/alex/.globus/userkey.pem"/>
                <cert-file value="/home/alex/.globus/usercert.pem"/>
        </credential>
        <GSISecureConversation>
                <privacy/>
        </GSISecureConversation>
</securityConfig>
```

Now the client can use this security descriptor to make an "authenticated" and encrypted communication and execute other methods that require authentication like VBEManager.updateVO, VBEManager.updateServiceProvider, etc.

2.2.3.4 *Server GSI Authenticated Communication*

In the services we need a security descriptor with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
        <credential>
                <key-file value="/etc/grid-security/containerkey.pem"/>
                <cert-file value="/etc/grid-security/containercert.pem"/>
        </credential>
```

```
        <auth-method>
                <GSISecureConversation
                        <protection-level>
                                <privacy/>
                        </protection-level>
                </GSISecureConversation>
        </auth-method>
        <authz value="gridtrust:org.gridtrust.util.GridTrustCertificatePIP"/>
</securityConfig>
```

The authentication method is set to GSISecureConversation or GSITrasport and the protection level (encryption) to privacy (encrypted and signed).

In the authorization configuration we use the GridTrustCertificatePIP Globus Policy Information Point to check the client certificate and store it in a SOAP message property.

2.2.4 *Installation on Globus*

VBE Manager is distributed as a Globus standard gar file that includes all the required libraries and can be downloaded from the Gridtrust Sourceforge web [5]. From the same url can be downloaded the Mysql script to create the VBE Manager database

2.2.5 *Configuration*

2.2.6 *Database configuration*

A Mysql server must be installed. In the file $GLOBUS_LOCATION/etc/org_gridtrust_vbe/jndi-config.xml must be set the following parameters in the resource "VBEDatabase":
url: Database URL. For example: jdbc:mysql://localhost:3306/vbe
username: Database Username
password: Database User password

2.2.7 *CA and GridTrust services configuration*

In the properties file $GLOBUS_LOCATION/etc/org_gridtrust_vbe/vbe.properties must be configured the CA certificate and private key and the URI to communicate with the others GridTrust services.

```
cakey: CA Private Key
caCert: CA Certificate
trsURI: TRS Service URI
ppmURI : PPM Service URI
srbURI: SRB Service URI
```

For example:

```
cakey = /home/globus/.globus/simpleCA/private/cakey.pem
caCert = /home/globus/.globus/simpleCA/cacert.pem
trsURI= http://130.37.193.48:9000/wsrf/services/TrustReputationService
ppmURI= http://130.37.193.48:9000/wsrf/services/gridtrust/ppm/PPManager
srb= http://130.37.193.48:9000/wsrf/services/gridtrust/srb/SRBManager
```

2.2.8 VBE as a service security config

In the security descriptor $GLOBUS_LOCATION/etc/org_gridtrust_vbe/security-config.xml must be configured the authentication and authorization of the VBE service with at least the GridTrustCertificatePIP Globus Policy Information Point. The following security descriptor, included in the gar file, should satisfy all common needs.

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
        <credential>
                <key-file value="/etc/grid-security/containerkey.pem"/>
                <cert-file value="/etc/grid-security/containercert.pem"/>
        </credential>
        <auth-method>
                <GSITransport>
                        <protection-level>
                                <privacy/>
                        </protection-level>
                        </GSITransport>
                <GSISecureConversation>
                        <protection-level>
                                <privacy/>
                        </protection-level>
                </GSISecureConversation>
        </auth-method>
         <authz value="gridtrust:org.gridtrust.util.GridTrustCertificatePIP"/>
</securityConfig>
```

2.2.9 VBE as a client of other services security config

In the security descriptor $GLOBUS_LOCATION/etc/org_gridtrust_vbe/client-security-config.xml must be configured the authentication and authorization to connect to the other GridTrust services like SRB or TRS. The following security descriptor, included in the gar file, should satisfy all common needs.

```
<securityConfig xmlns="http://www.globus.org">
        <credential>
                <key-file value="/etc/grid-security/containerkey.pem"/>
                <cert-file value="/etc/grid-security/containercert.pem"/>
        </credential>
        <GSISecureConversation>
                <privacy/>
        </GSISecureConversation>
</securityConfig>
```

2.2.10 Tomcat integration

2.2.10.1 Start/Stop script

In the Start/Stop script must be defined the variable GLOBUS_LOCATION and imported the globus-devel-env.sh script

```
#!/bin/bash
export JAVA_HOME="/usr/local/java"
export CATALINA_OPTS="-Xms512m -Xmx512m"
export GLOBUS_LOCATION="/usr/local/globus"
export PATH=/usr/local/java/bin:$PATH
```

```
export CATALINA_HOME=/usr/local/tomcat

case "$1" in
      'start')
            echo "Starting Tomcat..."
            source /usr/local/globus/etc/globus-devel-env.sh
            su tomcat -c "/usr/local/tomcat/bin/startup.sh"
      ;;
      'stop')
            echo "Stopping Tomcat..."
            su tomcat -c "/usr/local/tomcat/bin/shutdown.sh"
      ;;
esac
exit 0
```

2.2.10.2 Configuration

It's necessary to configure the Globus Connector and Valve in the server.xml configuration file.
In <Service name="Catalina"> it's necessary to add the following:

```
<Connector
   className="org.globus.tomcat.coyote.net.HTTPSConnector"
   port="8443" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   autoFlush="true" clientAuth="true"
   disableUploadTimeout="true" scheme="https"
   enableLookups="true" acceptCount="10" debug="0"
   protocolHandlerClassName="org.apache.coyote.http11.Http11Protocol"
   socketFactory="org.globus.tomcat.catalina.net.BaseHTTPSServerSocketFactory"
   cert="/etc/grid-security/tomcatcert.pem"
   key="/etc/grid-security/tomcatkey.pem"
   cacertdir="/etc/grid-security/certificates"
/>
```

and in <Engine name="Catalina" defaultHost="localhost"> the following:

```
<Valve className="org.globus.tomcat.coyote.valves.HTTPSValve55"/>
```

The user that runs Tomcat must have read permissions in the certificate and private key.

2.2.10.3 Required Libraries

It's necessary to copy the following Globus libraries to tomcat directories:

To $CATALINA_HOME/common/lib:
cog-axis.jar
cog-jglobus.jar
cog-url.jar
cryptix32.jar
cryptix-asn1.jar
jce-jdk13-131.jar
log4j-1.2.13.jar
puretls.jar

To $CATALINA_HOME/server/lib:
cog-tomcat.jar

2.2.10.4 *Browser configuration*

GridTrust CA certificate installation: The VBE CA certificate must be added to the browser trusted certificate authorities.

Figure 2 - PLEASE ADD A DESCRIPTION OF THE FIGURE HERE

GridTrust Client certificate installation: The client certificate received from the VBE and the private key must be packaged into a pkcs12 file container. This can be done with the GTUtil application (see section 2.2.12). The pkcs12 file must be imported in the browser user certificates.

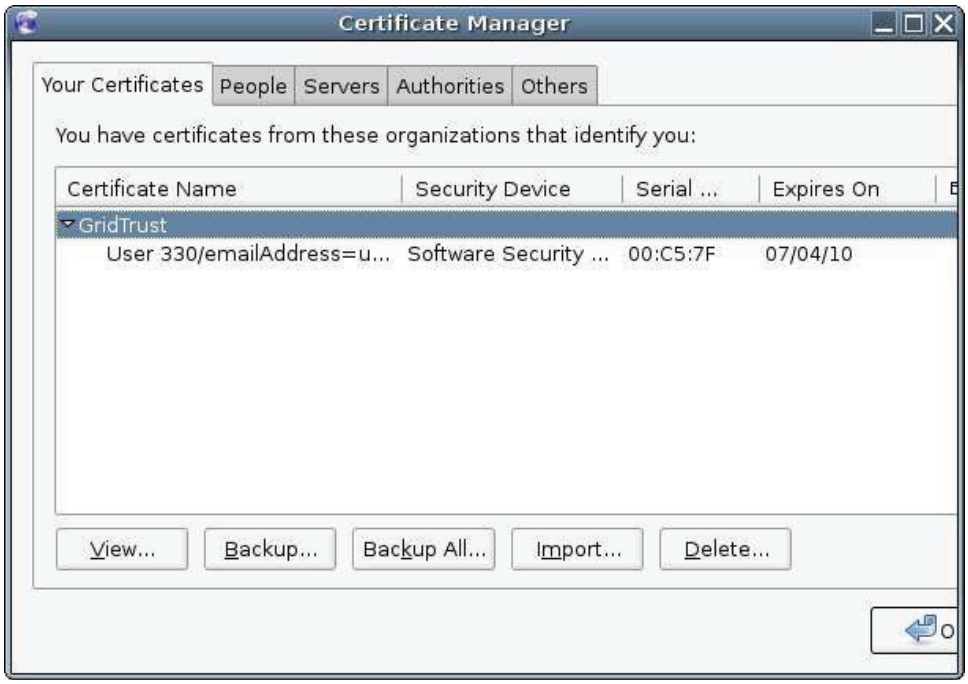

Figure 3 - PLEASE ADD A DESCRIPTION OF THE FIGURE HERE

2.2.11 VBE Access Portal

The VBE Portal is a web interface to the VBE Manager. Can be used as is or like a sample to make a custom portal for your organization.

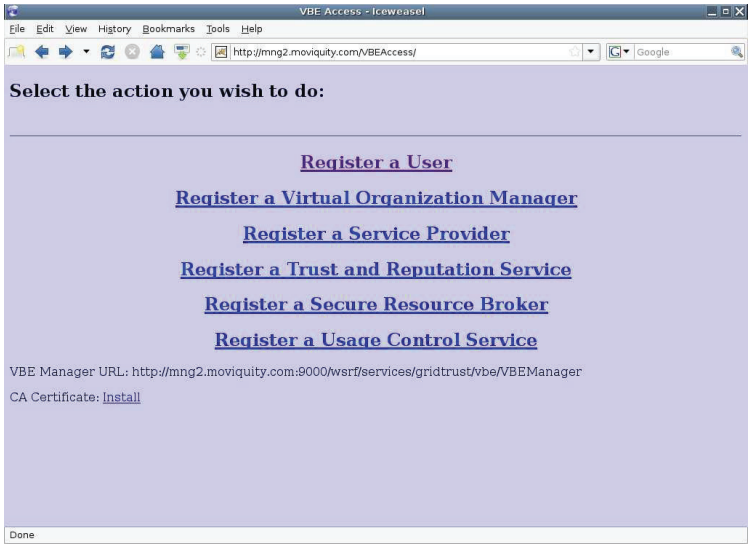

Figure 4 - PLEASE ADD A DESCRIPTION OF THE FIGURE HERE

2.2.11.1 Installation

VBE Access Portal can be downloaded from the GridTrust SourceForge repository [5]. Installation only requires uncompressing the package in a Tomcat or other container web app directory.

2.2.12 GTUtil application.

GTUtil is a Java client application that uses the GridTrust library util to create private and public keys, proxy certificates and export certificates to PKCS12 stores.

2.2.12.1 Installation

GTUtil compiled jar or source tar.gz/zip files and BouncyCastle library [12] jar (bcprov) can be downloaded from the GridTrust SourceForge repository [5].

The Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files [11] must be downloaded and copied to the lib/security directory of the Java virtual machine.

The BouncyCastle jar must be copied to the lib/ext directory of the Java virtual machine.
The following line must be added to the file lib/security/java.security of the Java virtual machine:

```
security.provider.7=org.bouncycastle.jce.provider.BouncyCastleProvider
```

2.2.12.2 Private and public key generation

The private key must be stored in a secure place, the public key should be used to register in the VBE Access portal or send to the VBE Access portal administrator for being registered in the VBE.

To generate a private and public key the user must select an output directory. In this directory will be saved the privatekey.pem and publickey.pem files.

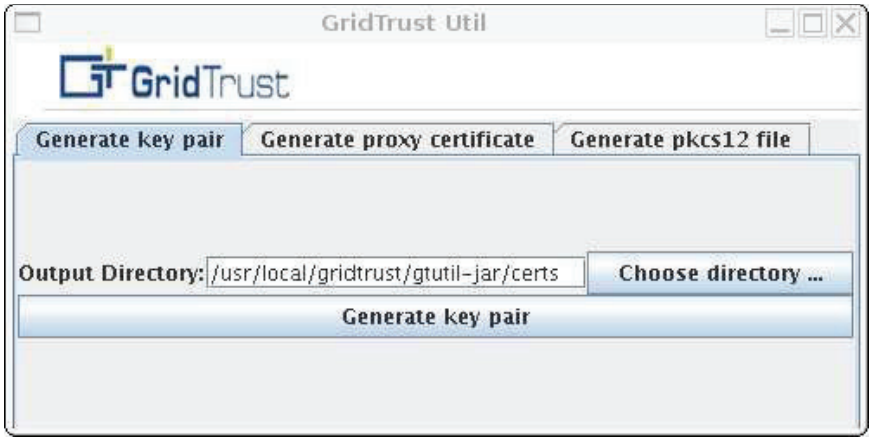

Figure 5 - PLEASE ADD A DESCRIPTION OF THE FIGURE HERE

2.2.12.3 Proxy certificate generation

Proxy certificate must be sent to the web administrator or VO Manager for identity delegation.

To generate a proxy certificate the user must select a VBE certificate, the corresponding

private key, an output directory to store the proxy certificate and the validity in days of the proxy certificate.

Figure 6 - PLEASE ADD A DESCRIPTION OF THE FIGURE HERE

2.2.12.4 PKCS12 file generation

PKCS12 file must be installed in the client browser or in a smart card.

The user must select a VBE certificate, the corresponding private key, an output directory to store the PKCS12 file and a password for the file.

Figure 7 - PLEASE ADD A DESCRIPTION OF THE FIGURE HERE

2.3 Usage Examples

2.3.1 Transporter Portal Demo

This demo is part of the Supply Chain Demo and shows the GridTrust features like VO management, SRB search, UCON policy, Tomcat/client browser integration security between service providers, etc. In this demo can be found samples of the use of the project libraries to create GridTrust client applications.

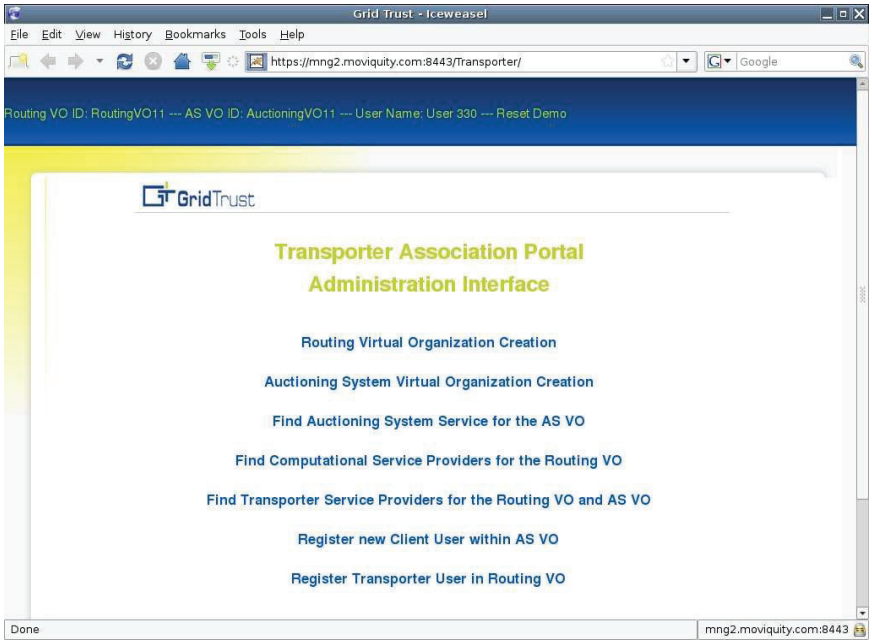

Figure 8 - PLEASE ADD A DESCRIPTION OF THE FIGURE HERE

2.3.1.1 Installation

Tomcat and web browser must be installed and configured according to instructions in sections 2.2.10 and 2.2.10.4. Transporter portal package can be downloaded from the GridTrust Sourceforge repository [5].

Transporter portal must be uncompressed in a Tomcat web application directory.
The properties file WEB-INF/classes/transporter.properties must be configured with the following parameters:

```
path: Directory to store application files  = /var/lib/transporter/
pathusercert: Subdirectory to store user proxy certificates and vo certificates
gramfolder = Subdirectory to store GRAM jobs files
host = Local host name
gridFTPPort = TCP port  that use the GridFTP server
gridFTPGroup = GridFTP group
globusLocation = Directory where Globus is installed
```

For example:

```
path = /var/lib/transporter/
pathusercert = usercerts/
gramfolder = gramfolder/
host = mng2.moviquity.com
gridFTPPort = 2811
gridFTPGroup = gridftp
globusLocation = /usr/local/gridtrust/globus/
```

The directories referenced by properties “path”, “pathusercert” and “gramfolder” must be created and the user that runs Tomcat must have write permissions in those directories and belong to the “gridFTPGroup”.

Globus user permissions must be granted in the file /etc/sudoers to the group "gridFTPGroup" to submit GRAM jobs.

```
globus ALL=(gridftp) NOPASSWD: /usr/local/globus/libexec/globus-gridmap-and-execute -g /etc/grid-security/grid-mapfile
/usr/local/globus/libexec/globus-job-manager-script.pl *
globus ALL=(gridftp) NOPASSWD: /usr/local/globus/libexec/globus-gridmap-and-execute -g /etc/grid-security/grid-mapfile
/usr/local/globus/libexec/globus-gram-local-proxy-tool *
```

2.3.2 *Auctioning System Demo*

This demo is part of the Supply Chain Demo and shows how to make GridTrust services with authentication, authorization, SRB search, job submissions, UCON security, etc. Must be installed the packages AuctioningSystem and TransporterService that can be downloaded from the GridTrust Sourceforge repository [5] in source form or in a gar file.

2.3.2.1 AuctioningSystem Installation

A client security descriptor in the path $GLOBUS_LOCATION/etc/org_gridtrust_as/client-security-config.xml must be configured according to instructions is section 2.2.3.3 (Client GSI Authenticated Communication) and a server security descriptor in the path $GLOBUS_LOCATION/etc/org_gridtrust_as/security-config.xml must be configured according to instructions in section 2.2.3.4 (Server GSI Authenticated Communication).
In the file $GLOBUS_LOCATION/etc/org_gridtrust_as/as.properties must be configured the following variables:

```
certificate: Certificate file path
privateKey: Private key file path
voCertsDirectory: Directory to store VO certificates
vbeURI: VBE URI
auctionIdPrefix: Auctioning System number
```

For example:

```
certificate = /etc/grid-security/auctioningsystem1.pem
privateKey = /etc/grid-security/privatekey.pem
voCertsDirectory = /var/lib/auctioningsystem1/vo-certificates/
vbeURI = http://mng2.moviquity.com:9000/wsrf/services/gridtrust/vbe/VBEManager
auctionIdPrefix = 1
```

Directory voCertsDirectory must be created and the user that runs Globus must have write permissions.

2.3.2.2 TransporterService Installation

A client and a server security descriptors must be configured like with the AuctioningSystem.
In the file $GLOBUS_LOCATION/etc/org_gridtrust_ts/ts.properties must be configured the following variables:

```
certificate: Certificate file path
privateKey = Private key file path
voCertsDirectory = Directory to store VO certificates
vbeURI = VBE URI
transporterType: Type of transporter (Good or Bad)
```

```
adminId = Administrator ID. Only de administrator can retrieve the transporter status.
```

For example:

```
certificate = /etc/grid-security/transporter1.pem
privateKey = /etc/grid-security/privatekey.pem
voCertsDirectory = /var/lib/transporterservice1/vo-certificates/
vbeURI = http://mng2.moviquity.com:9000/wsrf/services/gridtrust/vbe/VBEManager
transporterType = bad
adminId = user330@moviquity.com
```

In the file $GLOBUS_LOCATION/etc/org_gridtrust_ts/transporter.properties must be configured the variables related with the routing calculation.

2.3.3 Client and services development

VBE/VO/CA system have a complete and simple API to register users or service provider, manage VOs and certificates. API is divided in three parts.

- VBE API with all the methods to register, update and delete users, service providers, VOs and users and services providers in VOs.

- VO Library with a simple class VOManager that wraps all the VO related task and VO certificates management.

- GridTrust Library with help classes to manage certificates, private and public keys, submit jobs, etc.

2.3.3.1 VBE API Example - VBE User registration

In this example we execute the VBE registerUser method with a RegisterUser class that encapsulates the user information like name, email, organization, etc.

```
VBEManagerServiceAddressingLocator locator = new VBEManagerServiceAddressingLocator();
EndpointReferenceType endpoint = new EndpointReferenceType();
endpoint.setAddress(new Address(vbeURI));
VBEManagerPortType vbe = locator.getVBEManagerPortTypePort(endpoint);
String CLIENT_DESC = "/home/user/client-security-config.xml";
((Stub)vbe)._setProperty(Constants.CLIENT_DESCRIPTOR_FILE, CLIENT_DESC);
RegisterUser regUser = new RegisterUser();
regUser.setCommonName(valor);
regUser.setEmailAddress(valor);
regUser.setOrganization(valor);
regUser.setOrganizationalUnit(valor);
regUser.setCountry(valor);
regUser.setState(valor);
regUser.setLocality(valor);
RegisterUserResponse responseUser = vbe.registerUser(regUser);
GridTrustCertificate certificate = new GridTrustCertificate(responseUser.getCertificate());
Certificate.writeCertificateToFile(pathToCertificatefile);
```

2.3.3.2 *VO Library Example – VO registration*

In this example we register a new VO using the VO Library.

```
VOManager vom = new VOManager(vbeURI, securityDescriptorPath);
GridTrustCertificate  voCert = vom.registerVO(id, description, null);
voCert.writeCertificateToFile(localFilePath);
```

Security Descriptor Example:

```
<securityConfig xmlns="http://www.globus.org">
    <credential>
          <key-file value="/var/lib/transporter/privatekey.pem"/>
          <cert-file value="/var/lib/transporter/certificate.pem"/>
    </credential>
    <GSISecureConversation>
        <privacy/>
    </GSISecureConversation>
</securityConfig>
```

2.3.3.3 *VO Library Example – VO User Registration*

```
VOManager vom = new VOManager(vbeURI, securityDescriptor)
GridTrustCertificate voCert = new GridTrustCertificate(voOwnerCertificatePath);
GridTrustCertificate userCert = new GridTrustCertificate(userCertPath);
GridTrustCertificate voUserCert = vom.registerVOUser(voCert, userCert);
voUserCert.writeCertificateToFile(localFilePath);
```

2.3.3.4 *VO Library Example – VO Service Provider Registration*

```
VOManager vom = new VOManager(vbeURI, securityDescriptor)
GridTrustCertificate voCert = new GridTrustCertificate(voOwnerCertificatePath);
GridTrustCertificate serviceProviderCert = new GridTrustCertificate(serviceProviderCertPath);
GridTrustCertificate voServiceProviderCert = vom.registerVOServiceProvider(voCert, serviceProviderCert);
voServiceProviderCert.writeCertificateToFile(localFilePath);
```

2.3.3.5 *VO Library Example – VO User Deletion*

```
VOManager vom = new VOManager(vbeURI, securityDescriptor)
vom.deleteVOUser(voCert, userId);
```

2.3.3.6 *VO Library Example – VO Deletion*

```
VOManager vom = new VOManager(vbeURI, securityDescriptor)
vom.deleteVOU(voCert);
```

2.3.3.7 *GridTrust Library Example – Read client name from the client certificate in a GridTrust service*

If we have used the GridTrustCerticiatePIP Policy Information Point in the service security descriptor we can retrieve the client certificate with the following code:

```
GridTrustCertificate clientCert = GridTrustCertificate.getConnectionCertificate();
String username = clientCert.getName();
```

2.3.3.8 GridTrust Library Example – Read certificate from a file

```
GridTrustCertificate cert = new GridTrustCertificate(pathToFile);
```

2.3.3.9 GridTrust Library Example – Submit a job to a GRAM with UCON

GridTrustJobDescriptor and GridTrustJob classes integrates the VO, UCON and GRAM systems in a simple way to use. In this example we send a job to a GRAM service that uses UCON and retrieve the "uconViolation" and "uconLog" parameters.

```
GridTrustJob job = new GridTrustJob(tempFolder, gridFTPGroup, globusLocation);
String gramFolder = job.getDirectoryPath();
GridTrustJobDescriptor descriptor = new GridTrustJobDescriptor(host, gridFTPPort,  gramFolder,  classFileName,
arguments.split(","), classpath, inFileName, propFileName, outFileName, maxCpuTime, maxMem, stdout, stderr,  uconLog);
GridTrustCertificate voUserCertificate = new GridTrustCertificate(voUserCertificatePath);
GridTrustProxyCertificate userProxyCertificate = new GridTrustProxyCertificate(userProxyCertificatePath);
GridTrustProxyCertificate gramProxy = new GridTrustProxyCertificate(voUserCertificate,
userProxyCertificate.getKeyPair().getPrivate());
job.runJob(descriptor, gramProxy, gramURL);
String uconViolation = job.getUCONViolation();
String uconLog = job.getUCONLog();
```

3. TRUST AND REPUTATION SERVICE

3.1 Concepts and Functionalities

The Trust and Reputation service is one of the GridTrust Security Framework components. Its responsibility is to keep track of the past behaviour of users of a VO, and transform such usage history information in reputation credentials that can be considered by users, service providers and the GSF services when taking decisions.
The TR service will build reputation by collecting feedback from the Usage Control service, which monitors that users of the VO make appropriate usage of VO resources which are granted to them.
Security policies establish what 'appropriate usage' of VO resources is, and define expected behaviour a user should follow when working in the VO.

Model.
The reputation model implemented in this work is based on the approach described in [3], a utility based reputation system where the reputation of an entity is based on a utility function that reflects the satisfaction of a provider in relation to the consumer.
Users within a VO can execute a set of predefined actions against service providers, and should respect the security policies that protect such services. The C-UCON Monitor evaluates users in relation to their behaviour. If a user executes forbidden actions that break the security policies, he will receive a "bad action" feedback that would be reflected in his reputation. Conversely, if the user uses a service according to the policies, a "good action" is reported.
Therefore, the model gives rise to a mixed reputation system, where both good and bad actions are reported to the TRS, and are used as the information base to compute the level of trustworthiness of users.
To give an example closer to daily life, the model can be compared to a standard school mark system. The career of a student is consists of attending 5 school years, and his reputation is based on the mark he receives for each test in different subjects. Moreover the subject tests can be different for example there can be oral test, written test and so on.
Similarly, in the proposed reputation model, the reputation of a user is the result of "attending" or participating several VOs, and is based on the rates he receives for each action he executes. Different actions, likewise tests belongs to a specific subject in the school system, are grouped based on a common context, the Issue of Interest. Figure 9 shows an Entity Relationship (ER) diagram representing the building blocks of the reputation.

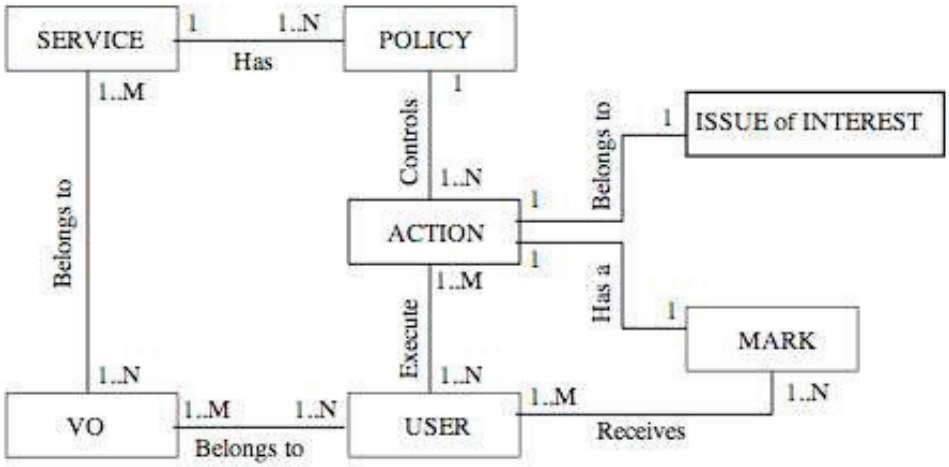

Figure 9 Trust Reputation Service - Logical Model

Reputation information is computed starting from feedbacks of the UCON service, and results from composition of the reports from all the UCON services.
Details on how reputation for a user is computed is provided in [3].

Given the goals of GridTrust project, the TR Service is oriented to monitoring compliance of user behaviour with respect to security policies, so it focuses on user reputation management. The model we adopted, as well as the implementation (with marginal changes), could be used to rate service providers as well.

VO reputation vs. VBE reputation.

In the GridTrust framework, collaboration among users and organizations happens in the context of Virtual Organizations (VO), which are temporary arrangements of users and resources, to achieve collaboration on a specific purpose, sometimes having a short lifetime. A Virtual Breeding Environment provides instead a long term context for collaboration among users and organizations.
Reputation information for a user can be stored at either levels, and transferred back and forth the levels. For example, when a new VO is setup or a new user registers with the VO, the application may want to initialize VO reputation for VO users from the value stored in the VBE. When a VO is dismantled, the reputation values for all users update the corresponding value in the VBE.
GridTrust uses a single TR Service at VBE level able to manage all the reputation for both VO Contexts and VBE context.

3.2 Architecture and Prototype description

3.2.1 System configuration

The released prototype has been developed in Java on Globus. The installation requires a full Globus Toolkit v4.0.x correctly installed and configured, and a MySQL server for the database. MySQL is not mandatory and other platforms can be used, anyhow TR Service by default is configured to use MySQL and only provides MySQL schema.

3.2.2 Installation on Globus

A full installation guide of the Trust and Reputation service the Trust and Reputation service is provided as attachment, which is also packaged with the code distribution and can be downloaded from the project repository [5].
The TR Service, like all the GridTrust project is hosted on sourceforge site and is released in Open Source under the Apache License V2.03. The latest release can be found in the download section of the site. The archive contains sources, dependencies library, WSDL definitions, database scripts and the service Grid Archive (GAR), that is the pre-compiled service containing all the files and information that the container needs to deploy a service.
Once downloaded the latest release and unpacked the archive, a number of steps are needed to get the TR Service operative.

- Configuration of the database
- Installing the TR Service on Globus from the GAR, or from the sources
- Testing the installation (by invoking the Web Service URL)

3.2.3 Design Details

The services introduced in GridTrust project are developed at the Grid middleware layer and in particular, since Globus [3] has been chosen as the reference Grid middleware, they are developed as Globus services.
The Globus Toolkit 4 (GT4) is based on standard Web Services architecture with the addendum of Web Services Resource Framework (WSRF), that specifies how to make a Web Services stateful.
Since reputation management is not a requirement of grid environments only, our implementation can potentially be used in other frameworks outside Globus e.g. in an Internet portal where there is the need to maintain users reputation, in a cloud computing platform etc.

3.3 Usage Examples

In this section we provide examples and suggestions on how to use the TRS in the development of an application.

3.3.1 Example - Distributed Content Management demonstrator

In the Distributed Content Management [D4.3] demonstrator of our project, the capabilities of Trust and Reputation Service are used as follows.

The Figure below shows the demonstrator architecture. The TRS is involved to maintain the reputation of users of a content management virtual organization (a collaboration platform allowing partner organizations to effectively share content in a scalable and secure way).
In this demonstrator, a policy enforcement and decision point is implemented at the VO level. Every time a client invokes a VO service that implies a protected action (for example, storing a content into the storage facility provided by the VO), trust and security policies are evaluated. Policies can be based on the current reputation of the user.

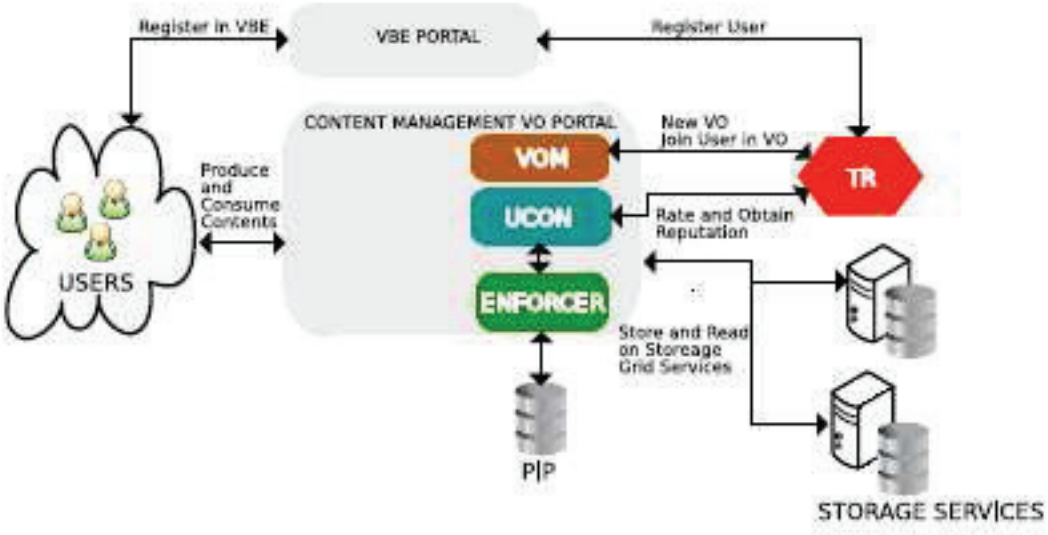

Figure 10 TRS in the Distributed Content Management demonstrator

For example, access may be denied if the user has a reputation lower than a certain threshold. Policies (more specifically, user policies) ensure that users invoke the service according to rules that have been agreed in advance, or must be anyway followed to use the service. For example, a user cannot store files into the VO storage service in excess of a threshold. If a violation of the policy occurs, a negative rating is sent to the TR. Also for each invocation of the protected action which does not break any security policy, the event is recorded and a positive rating is sent to the TR.

Using reputation information in policy decisions. Reading and updating user reputation is tipically a responsibility of the policy decision point.
In the demonstrator, a single VO-level policy enforcement and decision point is implemented. Every time a client invokes a VO service that implies a protected action (for example, storing content into the storage facility provided by the VO), trust and security policies are evaluated. Policies may be based on the current reputation of the user. For example, access may be denied if the user has a reputation lower than a certain threshold.

The task of rating a user action and reading user reputation while evaluating policies is performed by the Policy Enforcement Point of the application.
In the demonstrator, the PEP sends feedback (rate the user) after each invocation of a service, both positive and those that provoke a policy violation.

By using the strategy of rating each interaction of the user with the service, the reputation tends to stabilize to a value which keeps into account the 'average' behaviour of the user and a single event (either positive or negative) will not cause a dramatic change into the user reputation.
This is the recommended strategy to achieve a predictable behaviour of the reputation function.

3.3.2 Code Examples

Code snippets/ examples showing how to invoke the TRS from a client follow.

Client setup
The TR Service is exposed as Web Service and to interact with it you need the communication stubs. The chunk of code below shows how to obtain the Stubs.

```
// Create the request and response objects
AddVo request = new AddVo();
AddVoResponse response;

// Fill in the request
request.setVbeId(vbeId);
request.setVoId(voId);

try {
        // Call the service
        response = trsPort.addVo(request);

        // Verify the feedback
        if (!response.isOutcome()){
                // Something goes wrong!
                System.out.println(response.response.getMessage());
        }
} catch (RemoteException e) {
        //Handle exception
}
```

Initialization of a VO
When a VO is created, it must be registered with the TRS.

```
// Create the request and response objects
AddVo request = new AddVo();
AddVoResponse response;

// Fill in the request
request.setVbeId(vbeId);
request.setVoId(voId);

try {
        // Call the service
        response = trsPort.addVo(request);

        // Verify the feedback
        if (!response.isOutcome()){
                // Something goes wrong!
                System.out.println(response.response.getMessage());
```

```
        }
} catch (RemoteException e) {
        //Handle exception
}
```

Add a service to the VBE

For each service, the programmer is supposed to install some information in the VBE, more specifically, a descriptor for each action that will be reported to the TRS by the PEP when the user will use a service. In this example, we install 2 actions, one corresponding to the user using the service according to usage policies – which causes a positive feedback (+1), and one corresponding to a policy violation – which causes a negative feedback (-1) for the user.

In some scenarios the violation of different policies may require a different negative grade, in such cases several actions instead of only two (as shown in this example) would have been defined.

```
// Create the request and response objects
AddServiceInVbe request = new AddServiceInVbe();
AddServiceInVbeResponse response;

// Fill in the request
Service service = new Service();
Action action1 = new Action();
Action action2 = new Action();
//...

action1.setActionId("action1");
// 'good action' with a positive rating
action1.setRate(1);
action1.setIoi("ServiceUsage");

action2.setActionId("action2");
// bad action with negative rating
action2.setRate(-1);
action2.setIoi("ServiceUsage");
//...

Action[] actions = new Action[2];
actions[0] = action1;
actions[1] = action2;
//...

service.setActions(actions);
service.setServiceId(serviceId);

request.setService(service);
request.setVbeId(vbeId);

try {
        // Call the service
        response = trsPort.addServiceInVbe(request);

        // Verify the feedback
        if (!response.isOutcome()){
                // Something goes wrong!
                System.out.println(response.response.getMessage());
        }
} catch (RemoteException e) {
```

```
		//Handle exception
}
```

Add a user in the VBE

Each user must be registered to the VBE. For applications that create and shutdown Virtual Organizations frequently (such as the Supply chain demonstrator of the GridTrust project), this is done only once.

```
// Create the request and response objects
AddUserInVbe request = new AddUserInVbe();
AddUserInVbeResponse response;

// Fill in the request
User user = new User();
user.setUserId(userId);
request.setUser(user);
request.setVbeId(vbeId);

try {
		// Call the service
		response = trsPort.addUserInVbe(request);

		// Verify the feedback
		if (!response.isOutcome()){
			// Something goes wrong!
			System.out.println(response.response.getMessage());
		}
} catch (RemoteException e) {
		//Handle exception
}
```

Register a service in a VO and registering users

When creating a VO, it is required to register users and services to the VO. When registering users with the VO, it may be desirable to copy existing reputation information at the VBE level into the new VO. This may be important depending on the scenario. For example in the Content Management scenario, the VO is created once and is supposed to live for a long time (as long as the business relationships among the VO members exists); hence the case for transferring reputation information from the VBE is not relevant. Reputation of users in the VO will start from an <unknown> value. In another application, where a VO is a temporary setup to achieve a goal, may be desirable to save the reputation in the VBE and restore it for each new VO that is setup (user reputation should survive across different VOs).

```
// Register a service to the VO
// Create the request and response objects
RegisterServiceInVo request = new RegisterServiceInVo();
RegisterServiceInVoResponse response;

// Fill in the request
request.setServiceId(serviceId);
request.setVbeId(vbeId);
request.setVoId(voId);

try {
```

```
		// Call the service
		response = trsPort.registerServiceInVo(request);

		// Verify the feedback
		if (!response.isOutcome()){
			// Something goes wrong!
			System.out.println(response.response.getMessage());
		}
} catch (RemoteException e) {
		//Handle exception
}

// Register a user in a VO
// Create the request and response objects
RegisterUserInVo request = new RegisterUserInVo();
RegisterUserInVoResponse response;

// Fill in the request
request.setUserId(userId);
request.setVbeId(vbeId);
request.setVoId(voId);

try {
		// Call the service
		response = trsPort.registerUserInVo(request);

		// Verify the feedback
		if (!response.isOutcome()){
			// Something goes wrong!
			System.out.println(response.response.getMessage());
		}
} catch (RemoteException e) {
		//Handle exception
}
```

Rate a user

After all the initialization steps, this is the real task the TRS must accomplish... Rating a user is invoked typically by a PEP after service invocation.

```
// Create the request and response objects
RateUser request = new RateUser();
RateUserResponse response;

// Fill in the request
Request.setActionId("action1");
request.setServiceId(serviceId);
request.setUserId(userId);
request.setVbeId(vbeId);
request.setVoId(voId);

try {
		// Call the service
		response = trsPort.rateUser(request);
		// Verify the feedback
		if (!response.isOutcome()){
			// Something goes wrong!
			System.out.println(response.response.getMessage());
		}
} catch (RemoteException e) {
```

```
		//Handle exception
}
```

Obtain the user reputation

Similarly, obtaining the user reputation is the second important functionality of the TRS.

```
// Create the request and response objects
ObtainUserReputation request = new ObtainUserReputation();
ObtainUserReputationResponse response;

// Fill in the request
request.setUserId(userId);
request.setVbeId(vbeId);
request.setVoId(voId); // Set to null to obtain the VBE reputation instead

try {
		// Call the service
		response = trsPort.obtainUserReputation(request);

		// Verify the feedback
		if (!response.isOutcome()){
			// Something goes wrong!
			System.out.println(response.response.getMessage());
		}else{
			// Positive feedback!
			System.out.println(response.getReputation());
		}
} catch (RemoteException e) {
		//handle exception
}
```

Shut down a VO

Last, when a VO is terminated, the client should take care of unregistering the VO from the TRS and update reputation at the VBE level by incorporating the reputation gained by the user in the VO.

```
// Create the request and response objects
TerminateVo request = new TerminateVo();
TerminateVoResponse response;

// Fill in the request
request.setVbeId(vbeId);
request.setVoId(voId);

try {
		// Call the service
		response = trsPort.terminateVo(request);

		// Verify the feedback
		if (!response.isOutcome()){
			// Something goes wrong!
			System.out.println(response.response.getMessage());
		}
} catch (RemoteException e) {
		//Handle exception
}
```

3.3.3 *Example – The test client*

In the software distribution on SourceForge a test client is available which shows practical examples of how to invoke the service functionality.

4. REFERENCES

[1] GridTrust Framework Description Document – Internal GridTrust document
[2] OGSI specification: http://www.ggf.org/documents/GFD.15.pdf
[3] REPUTATION MANAGEMENT IN GRID-BASED VIRTUAL ORGANISATIONS Alvaro Arenas, Benjamin Aziz e-Science Centre, STFC Rutherford Appleton Laboratory, Oxfordshire, UK; Gheorghe Cosmin Silaghi Babes-Bolyai University, Faculty of Economics, Cluj-Napoca, Romania. Available online at http://epubs.cclrc.ac.uk/bitstream/3402/ReputationGrids.pdf
[5] GridTrust SourceForge project repository: http://sourceforge.net/projects/gridtrust/
[6] X509 Certificates RFC: http://www.ietf.org/rfc/rfc2459.txt
[7] Globus Simple CA: http://www.globus.org/toolkit/docs/4.0/security/simpleca/
[8] Globus Quick Start Guide:
http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html
[9] Globus GSI: http://www.globus.org/security/overview.html
[10] Globus Security Descriptors:
http://globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html
[11] Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files:
https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jce_policy-1.5.0-oth-JPR@CDS-CDS_Developer
[12] BouncyCastle library: http://www.bouncycastle.org/java.html