VOMS Core Services

EUROPEAN MIDDLEWARE INITIATIVE

VOMS 2.0 USER'S AND ADMINISTRATOR'S GUIDE

Document version:	**2.0.0**
EMI Component Version:	**2.0**
Date:	**April 19, 2011**

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-261611.

This work is co-funded by the EC EGEE project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-031688.

CONTENTS

Part I

INTRODUCTION

1 INTRODUCTION

This guide will explain how to install, setup and use the VOMS server and its associated client-side utilities. It will also describe how to use the provided API. This guide refers to VOMS 1.8.4 or greater.

Throughout this whole document, you will find sections marked thus:

Compatibility

Some information

These section contain informations regarding both back and forward compatibility between different versions of the applications and the APIs.

1.1 SERVICE ARCHITECTURE

The service follows an established client-server architecture. It consists of a server (vomsd), a client (voms-proxy-init) and some ancillary utilities (voms-proxy-info, voms-proxy-destroy, voms-proxy-list). After the service has been setup, users are supposed to replace use of grid-proxy-init with use of voms-proxy-init, generating a proxy certificate that while backward-compatible with the one generated by the globus command, contains extra informations about the user and the VOs he belongs to.

1.2 INTERACTIONS WITH OTHER SERVICES

There is no direct interaction with other services. Services that wish to take advantage of VOMS, must do so through the API described in a later section.

2 BASIC CONCEPTS

2.1 ATTRIBUTE CERTIFICATE

An Attribute Certificate (AC for short) is a PKI container, defined in RFC 3281[1], capable of containing a set of attributes tied to a specific identity. It is the system used by VOMS to issue its attributes.

2.2 GROUPS, ROLES AND GENERIC ATTRIBUTES

Members of a Virtual Organization can be organized into groups. These groups can then be directly represented as voms groups. Groups are organized in a hierarchical tree, where each group may have zero, one or more subgroups, with no limitation on tree depth. The root of the tree is fixed, and is the VO itself.

A group name contains the representation of the path leading to it from the root. For example, if a user were member of the subgroup `bologna` of the group `italian` in the VO `test`, the group name as represented by VOMS will be `/test/italian/bologna`.

Groups There are no effective limits on the length of a group name, excepts those enforced by the underlying DB in which the name is stored. However, only alphanumeric characters plus '-', '_' and '.' are allowed in group name.

Finally, membership in a subgroup requires membership in its parent group. From this it should be evident that all users must at least be members of the root group.

Roles Not all members of a group are necessarily equal, but some may occasionally have some special rights that, while not always needed, may indeed be necessary for the execution of special tasks. A simple example of this is the `softwaremanager` or `sgm` role that corresponds to the right of installing software on grid nodes.

This need is represented by the VOMS Roles. A VOMS Role is always associated to a specific group. E.g, holding the role `sgm` in the `/test/italian` group is a different thing than holding the same role in the `/test` group.

Restriction on role names are the same as restriction on group names. Furthermore, Role names starting with `VOMS` are reserved for use by VOMS itself. Finally, the special name `NULL` may indicate that no specific role is held.

Generic Attributes Finally, it should be noted that not all the charateristics of a user can be represented by a combination of groups and roles. For example, consider the need for registering the guarantor of a user.

For these cases, generic attributes are used. They consist of triple (name, value, qualifier), with the qualifier optional. As an example, the guarantor case above could be represented by the following tuple: (guarantor, "Guarantor Smith")

There are no hardcoded limits on the number, length and character usable in generic attributes.

2.2.1 FQAN

A Fully Qualified Attribute Name (FQAN for short) is a compact way to represent a user's membership in a group, along with its role holdership, if any.

Its general syntax would be: <groupname> [/Role=<rolename>]

For example, belonging to the group `/test/italian` may be represented by the following FQAN:

```
/test/italian
```

while holding the role `sgm` in the same group will be represented by the following FQAN:

```
/test/italian/Role=sgm
```

Compatibility

Older versions of VOMS used to append a `/Role=NULL` to the FQAN if no role was specified, and always added a `/Capability=NULL` at the end. This behaviour is now deprecated, and applications should move away from it. The option `-shortfqans` is available to generate the new format.

Accordingly, all examples in this guide will show the short format of FQANs.

Part II

System Administration Guide

3 COMPATIBILITY

With version 1.6.0 and onwards, compatibility with VOMS version 1.1.x and previous version has been dropped. This means that servers that are not capable of generating ACs are now unsupported.

4 INSTALLING AND RUNNING A VOMS SERVER

To complete configuration of `voms`, you are supposed to execute the `/usr/libexec/voms_install_db` command. It takes the following options:

–mysql-home This option lets you specify the home directory of mysql. This information is usually included in the $MYSQL_HOME environment variable, and if that is the case on your machine then you do not need to specify this option. For this reason, this option is only needed id MySQL support is desired.

–oracle-home This option specifies where the Oracle installation is based. For this reason, it should only be specified if Oracle support is desired. There are no defaults.

–db This is the name of the database that will contain the information about the VO. Its default name is "voms_<vo name>".

–port This is the port number where the VOMS server will be listening. The default is 15000, and recommended choices for servers other then the first are 15001, 15002, etc. . .

–voms-vo This is the name of the VO to which the VOMS server belongs. There is no default. For this reason, this option must be *always* specified.

–db-admin This is the name of the DB user which will create the tables. It is needed because the script needs to create a new DB and a new user. Its default value is "root", which is the standard on the default MySQL installation.

–db-pwd This is the password of the DB account specified by the previous option. There are no defaults. This is a required option.

–voms-name This is the username of the voms MySQL account that will be setup to access the newly created DB. Its default value is "voms_[VONAME]".

–voms-pwd This is the password associated with the *voms-name* account. If not specified, a random password is created. You should always specify a new value.

–code	This is a unique code for each server installed on the same host. It is a value between 0 and 65535, and its default is the value of **–port**. This option is DEPRECATED.

Compatibility

This option is only present for backwards compatibility with previous versions. Its default value is perfectly acceptable, and so should never be explicitly specified anymore.

–db-type	This specifies the type of db that will be used by the server. Currently accepted values are *mysql* and *oracle*. There is no default for this option. This is a required option.
–sqlloc	This specifies the name of the DB interface library. Again, there is no default for this option. The library should be in $LD_LIBRARY_PATH or in `ld.so.conf`, or the full path name should be specified. This is a required option.
–compat	This option must be specified if you plan to use voms 1.5.x on a MySQL backend with an old version of voms-admin. It requires –db-type to be *mysql*.
–newformat	This forces the server to generate ACs in the new (correct) format. This is meant as a compatibility feature to ease migration while the servers upgrade to the new version.
–socktimeout	This option specifies, in seconds, the maximum amount of time after which a stalled connection will be closed by the server.
–loglevel **–logtype** **–logformat** **–logdateformat**	These four options setup the logging mechanism. For an explanation of their meaning, see the definition at 5.2.1
–help	Prints a short reminder of these options.

A couple of example invocations follows: for the first VO,
voms-install-db –vo-name my-vo –mysql-pwd 'some' –voms-pwd 'thing'
For a second VO on the same host,
voms-install-db –db new-vo –port 15001 –vo-name new-vo –mysql-pwd 'some' –voms-name 'voms2' –voms-pwd 'thing'

The server also needs to have a host certificate installed. Obtain it from your CA using the CA-specific procedures, and then copy the certificate in `/etc/grid-security/hostcert.pem` and the private key to `/etc/grid-security/hostkey.pem`. The owners should be set to root.root for both files, and permission should be, respectively, 644 and 600 or, better, 444 and 400.

5 VOMS SERVER COMMAND LINE REFERENCE

Installing the server using the above described procedure should correctly create a set of configuration files that will execute it with the proper options. However, there are many other options that are not used by the default configuration script. The following lines will so describe the totality of the options.

5.1 SOFTWARE REQUIREMENTS

Running a VOMS servers poses a few requirements on the physical host: aside from the usual C/C++ libraries, expat 1.95 or later must be installed, along with C client libraries for either MySQL or Oracle. In the former case, at least version 4.0.10 is required, while for the latter 10.1 is the absolute minimum. Note that due to known bugs, MySQL 4.1.23 and MySQL 4.1.24 are not supported. MySQL version 5 is supported.

5.2 COMMAND LINE INTERFACE

–port	The port number on which the server should be listening. The default value is 50000
–vo	The name of the VO to which this server will belong. The default value is "unspecified".
–globus **–globusid** **–globuspwd**	These three options are OBSOLETE, and their value will be ignored.
–x509_cert_dir	The location where the CA certificates are kept. The default value is /etc/grid-security/certificates
–x509_cert_file	A file containing all the CA certificates. There is no default value.
–x509_user_proxy	The location of the server's proxy. There is no default value. Usage of this option is strongly discouraged.
–x509_user_cert	The location of the server's certificate. The default value is "/etc/grid-security/hostcert.pem"
–x509_user_key	The location of the server's private key. The default value is "/etc/grid-security/hostkey.pem"
–desired_name	OBSOLETE. This option will be removed in the future. Do *not* use it.
–foreground	OBSOLETE. This option will be removed in the future. Do *not* use it.

–username	The name of the user with which VOMS will access the DB. The default value is "voms"
–dbname	The name of the DB that VOMS will use. The default value is "voms".
–timeout	The maximum length of validity of the ACs that VOMS will grant. (in seconds) The default value is 24 hours
–passfile	The location of the file containing the password needed to access the DB. This file should be owned by root and have permissions set to 400. There is no default value. If this option is not specified, than the password will be asked to the user during server startup.
–uri	The URI that the server will publish for himself. The default value is <hostname>:<port>.
–version	Prints the version number and compilation date and then exits.
–backlog	Sets the backlog on the socket. The default value is 50.
–debug	Slightly modifies the internal workings of the server to ease debug. *Never* use it on production servers. Use of this option is guaranteed to severely hurt scalability, and should not be specified on a production server.
–conf	Lets you specify a file from which options will be loaded. This file should have exactly one option per line, and option that do have values should be specified in the format "option=value".
–code	OBSOLETE. This option will be removed in the future. Do *not* use it.
–sqlloc	Specifies the location of the DB interface library. If this is not in the library search path, than the full path should be specified. This is a required option.
–compat	Specify for compatibility with databases created by VOMS 1.5.4 or before. Only makes sense if the underlying DB is MySQL.
–socktimeout	Specifies the timeout after which an inactive connection should be dropped. Defaults to 60 seconds.
–contactstring	Specified the hostname for the DB.
–mysql-port	Specifies the port where MySQL is listening.
–mysql-socket	Specifies the socket where MySQL is listening.
–newformat	Creates AC in the newer, correct format. Warning! Specifying this option breaks compatibility with VOMS APIs version 1.6.8 or before.
–logfile	
–loglevel	
–logdateformat	

–logformat	
–logtype	
–logmax	These six options are explained below in the Logging Options section.
–skipcacheck	This option disables checking the CA of the authenticated user against his CA as registered into the VOMS DB. As such, two different certificates, issued by two different CAs but having the same subject will become indistinguishable to VOMS. This may be useful during CA transitions, but in general it is a better idea for the user to reregister with the new certificate.
–shortfqans	This option activates the generation of shorter FQANs, with the `/xx=NULL` part removed.
–syslog	This option activates logging onto syslog.
–base64	This option activates the usage of the OpenSSL variant of base64 as the default encoding of the server's answers. Note that a client may force the use of this version even if the option is not specified.
–help	
–usage	These two options are synonymous and will print a short usage reminder and then quit.
–nologfile	Do not use the application-specific logfile.

5.2.1 LOGGING OPTIONS

Logging in VOMS is highly configurable both in its format and content. For this reason, an entire section is dedicated to it.

First of all, the **–logfile** option specifies where the main logfile is placed. If not specifiedm this defaults to "/var/log/<voname>.log".

However this is just the main file. When the amount specified by the **–logmax** option is reached, the existing logfile is renamed by prepending `.1` to its name and a new one is created. If the `.1` file already existed, than it is renamed as a `.2`, and so on and on.

Then, the **–loglevel** option specified how much to log. It takes parameter according to the following table:

1	LEV_NONE	Do not log
2	LEV_ERROR	Only log error messages.
3	LEV_WARN	Also warn error messages.
4	LEV_INFO	Also log informational message.
>=5	LEV_DEBUG	Also log debugging information.

Where each successive level also logs all information that would be logged by the previous ones. The default value is 4, (LEV_INFO). Please note the specifying LEV_DEBUG more than quintuples the amount of information logged.

The option **–logtype** decides what type of information to log. Its value is decided by OR-ing values from the following table:

1	T_STARTUP	Startup messages.
2	T_REQUEST	Request handling.
4	T_RESULT	Result handling.

The default value is 7, i.e: (T_REQUEST | T_STARTUP | T_RESULT). It is strongly suggested that this value is left as it is.

5.3 SERVER SETUP AND MAINTENANCE

Aside from the setup specified by the above command line options, the following configuration needs to be done:

- a host certificate/host ket couple needs to be installed, along with a directory containing all acceptable CAs along with the corresponding CRL.
- if the underlying DB is meant to be Oracle, a `oranames.tns` file should be available to detail the connection. Such a file should either be referenced by the `$TNS_ADMIN` variable or be placed in the `/etc/voms/<voname>/` directory. The content and syntax of this file is out of the scope of this document. See Oracle documentation for details.

While the server is running, it is possible to change the value of most of his options without requiring a server restart. If the options were loaded from a `.conf` file, then after changing it it is usually enough to send an (HANGUP) signal (via `kill -HUP`) to the VOMS process with the higher `PID`.

This is not valid for the following options, which *do* require a restart when changed: `-port, -backlog, -conf, -sqlloc` and, if the underlying DB is Oracle, the following options also require a restart when changed: `-dbname, -username, -passfile` and the content of the pointed file.

5.4 EXAMPLE CONFIGURATION

The following example configuration is taken from the dteam production server, with passwords and usernames obviously changed.

- Content of `/etc/voms/dteam`

```
-rw-r-----    1 root     root          228 Jun  8 16:55 voms.conf
-rw-r-----    1 root     root           13 Jun  8 16:53 voms.pass
```

- Content of `/etc/voms/dteam/voms.conf`

```
--code=15004
--dbname=prod
--logfile=/var/log/glite/voms.dteam
--loglevel=4
--logtype=7
--passfile=/etc/voms/dteam/voms.pass
--port=15004
```

```
--sqlloc=/usr/lib/libvomsoracle.so
--username=voms_user
--vo=dteam
```

- Content of `/etc/voms/dteam/voms.pass`

```
LunReC@f
```

- Content of `$TNS_ADMIN`

```
prod=
   (DESCRIPTION =
     (ADDRESS = (PROTOCOL = TCP)(HOST = lcgr4-v.cern.ch)(PORT = 10121))
     (ADDRESS = (PROTOCOL = TCP)(HOST = lcgr1-v.cern.ch)(PORT = 10121))
     (ADDRESS = (PROTOCOL = TCP)(HOST = lcgr2-v.cern.ch)(PORT = 10121))
     (ADDRESS = (PROTOCOL = TCP)(HOST = lcgr3-v.cern.ch)(PORT = 10121))
     (LOAD_BALANCE = yes)
     (CONNECT_DATA =
       (SERVER = DEDICATED)
       (SERVICE_NAME = lcg_voms.cern.ch)
       (FAILOVER_MODE = (TYPE = SELECT)(METHOD = BASIC)(RETRIES = 200)(DELAY = 15))
     )
   )
```

6 HOST SETUP

There is some common setup that must be done prior to use of either the client commands or the APIs.

6.1 VOMSDIR SETUP

The X509_VOMS_DIR environment variable must point to a directory where a copy of all the certificates of the supported VOMS servers are kept. In this directory there should be a subdirectory for each VO, named as the VO itseld, and all certificates belonging to a server for that VO should be placed in the subdir.

Alternatively, a file named `<hostname>.lsc` can be placed in the directory. If so, its format should ba a list of subject/issuer DN couples detailing the exact certification path, up to and including the CA, of a certificate authorized to sign ACs for the specific VO. More then one such path can be specified by separating the paths with a '—— NEXT CHAIN ——' line.

If the X509_VOMS_DIR variable is not specified or is empty, than it defaults to "/etc/grid-security/vomsdir".

In case both the .lsc and the cartificates are present in a directory, the .lsc file supercedes the certificates.

Compatibility

As a backwards compatibility feature, if a server certificate is not found in the $X509_VOMS_DIR/<VO> directory, then it is searched for also in $X509_VOMS_DIR. This is temporary, and will eventually be removed, so do not rely on this working.

6.2 VOMSES FILES

This is only necessary if it is foreseen that there will be programs installed that will need to contact a VOMS server, like voms-proxy-init.

Be sure that in "/etc/vomses" or in "$HOME/.voms/vomses" you have put a copy of the *vomses* file distributed by all the VOMS servers you wish to contact. This subtree will be recursed into to examine all pertinent files.

The format of a vomses file is a set of lines, each with the following format: "nick" "hostname" "port" "server's DN" "alias" "globus version"

Where `nick` is the nickname which will be used as the VO name for the voms-proxy-init command, `hostname` is the host on which the server is running, `port` is the port on which it is listening, `server's DN` is the DN of the server's certificate, `alias` is ignored for now, but should be put equal to `nick`, and finally `globus version` is optional, and unnecessary if the server is running voms server 2.0 or later, but to keep compatibility with VOMS servers <= 1.8 it should encoded, in the same way as for the `-globus` option of voms-proxy-init, the version of globus on which the server is running.

It is possible to assign the same nick to several servers. This is interpreted by voms-proxy-init as declaring the servers as replicas of each other, in which case they will be contacted in random order until one succeeds or all fail.

The easier way to comply to both previous points is to install the VO config RPM that should be distributed by the VOMS servers themselves.

7 REPLICATING VOMS

The VOMS command-line utilities come with builtin support for replicated VOMS servers. Once two or more vomses files are installed declaring multiple servers as belonging to the same VO, voms-proxy-init and voms-proxy-list consider those servers as replicas of each other, and when operations regarding that VO are requested, all replicas are tried in random order until the request succeed for one of them or all of them reported a failure.

Setting two or more servers as replica is as simple as replicating the underlying DB and then point the new VOMS server to the replica. What follows is detailed instruction on how to do so for MySQL, also referring to the two provided scripts, voms_replica_master_setup.sh and voms_replica_slave_setup.sh. Instructions for Oracle will follow in a subsequent version of this guide.

7.1 MYSQL REPLICATION

Replication of the MySQL server is supported for all versions of MySQL which are explicitly supported by the server. Note however that MySQL master and slave versions cannot be freel mixed. In particular a

version 4 MySQL *cannot* be setup as a version 5 MySQL slave. MySQL itself refuses such a setup. The reverse however works.

The first step is configuring the master. This is done by running on it the voms_replica_master_setup.sh script, which will retrieve a set of informations needed to properly setup a slave, and also change the server's configuration so that it can act as a master.

> Compatibility
>
> Note that this script assumes it can scrap the existing server configuration. If this is not acceptable, please read section 7.1.1 to discover what it does, and apply the changes yourself.

A typical execution of the voms_replica_master_setup.sh script may be the following:

```
[marotta@testbed002]$ voms_replica_master_setup.sh --mysql-pwd=password \
  --slave-host voms-slave.cnaf.infn.it --master-db=voms_testvo \
  --mysql-version=5 --
```

And you can expect output like this:

```
Send these informations to the administrator of the slave server:
Log File    : testbed002-bin.000030
Log Position: 94
Account name: replica
Account pwd : gter!fgi
DB name     : voms_testvo
Ignore      :

Also, send this file: voms_testvo.dump
```

It is possible that the first two fields will be empty. In this case, the values should be '' (the empty string) and `4`.

Note that if you are using MySQL 5, then you must ensure that no other application is updating the VOMS DB while running the script. Alternatively, you may rerun a step by hand (see 7.1.1) and update the results above before sending them to the slave server administrator.

Now, send these values to the administrator of the slave server.

On the slave server, copy voms_testvo.dump on the directory from which you are going to call voms_replica_slave_setup.sh and call the script.

> Compatibility
>
> Note that this script assumes it can scrap the existing server configuration. If this is not acceptable, please read section 7.1.1 to discover what it does, and apply the changes yourself.

A typical invocation can be this:

```
[marotta@datatag6]$ voms_replica_slave_setup.sh --mysql-pwd=password \
  --replica-user=replica --replica-pwd=gter!fgi \
  --master-host testbed002.cnaf.infn.it --master-db=voms_testvo \
  --log-file=testbed002-bin.000030 --log-file-position=94
```

If it is successfull you should see MySQL being stopped and restarted.

Replication is now setup. See the next subsection for details on what happened under the hood.

Replication protection with SSL is supported. To activate it, the master and the server should specify the `-require-ssl` and `-use-ssl` options respectively. Note however that not all distributions of MySQL support SSL. A quick way to discover wether the distribution you use does is '`mysql -ssl`'. If SSL is not supported, mysql will complain that it does not understand the option.

7.1.1 MySQL Replication – Under the Hood

Master setup

First of all, a new user for replication is created and replication slave rights granted to it. This rights must be granted on all tables of all databases, regardless of the specific database you are trying to replicate, because MySQL will not accept the command otherwise.

Then, tables are flushed and locked, and the master status obtained. This corresponds to the `Log File` and `Log File Position` fields. The next step, before releasing the lock, is to obtain a dump of the current state of the DB. Note that MySQL 5 does not allow to do this from the same command, and so it is done before the lock. Alternatively, and better, the master admin should lock the system by hand, and then generate the dump by hand from another shell. If so, please be sure to update the log informations before sending them to the slave.

As the next step, the DB configuration file, `my.cnf` is edited. The values of the `datadir`, `socket` and `old_password` options, if present, are preserved, but everything else is created anew.

Specifically, the server is set to output logs in binary mode (`log-bin`), a server id is set and put to `1` (`server-id`), and then all write statements are immediately synchronized in binlog (`sync_binlog=1`).

Some additional options related to safe writing in binlog are then added for MySQL4, specifically `innodb-safe-binlog` and `innodb_flush_log_at_trx_commit=1`.

If SSL is required, then the corresponding options, `ssl`, `ssl-capath`, `ssl-cert`, `ssl-key` are then set.

This ends the server reconfiguration. At this point the server is restarted, and it is now ready to act as a Master.

Slave setup

First of all MySQL is stopped and the `my.cnf` rewritten. the `log-bin` and `server-id` options are set. The argument to `server-id` should be greater than 1 and unique among all slaves.

If SSL is required, then the corresponding options, `ssl`, `ssl-capath`, `ssl-cert`, `ssl-key` are then set.

`replicate-do-db` is set to indicate which database to replicate. If the previous versions of `my.cnf` already replicated some databases, those values are preserved.

Finally, MySQL is restarted and it is told how to contact the slave.

Part III
User's Quickstart

8 USING THE CLIENT COMMAND

8.0.2 USER CREDENTIALS

While user credentials may be put anywhere, and then their location passed to voms-proxy-init via the appropriate options, there are obviously default values.

User credentials should be posted in the `.globus` subdirectory. Both PKCS12 and PEM formatted credentials are okay. The default name for the PKCS12 are usercert.p12 or usercred.p12, while usercert.pem and userkey.pem are the default names for the PEM formatted one.

In case both the PEM and PKCS12 formats are present, PEM takes precedence.

The user certificate should at the most have permission 600, while the user key shoud be 400.

8.1 USING THE CLIENT COMMANDS

What follows here is an explanation of all the client commands, along with some examples of their usage. It is *not* meant to be a complete reference. For this see the User's Guide section of this document.

8.1.1 VOMS-PROXY-INIT

voms-proxy-init is the command that should be used to create a proxy for usage on the grid. Its basic syntax is:

```
voms-proxy-init --voms valerio
```

where voname is the name of the VO to which the user belongs. This will create a proxy containing all the groups to which the user belongs, but none of the roles. Also, the `-voms` option may be specified multiple times in cas the user belongs to more than one VO.

It is also possible to omit the –voms option entirely. This will however result in the creation of a completely globus-standard proxy, and is not advised since such proxies will not be usable under gLite 3.0.0 and beyond.

As stated above, no roles are ever include in proxy by default. In case they are needed, they must be explicitly requested. For example, to request the role `sgm` in the `/test/italian` group, the following syntax should be used:

```
voms-proxy-init --voms test:/test/italian/Role=sgm
```

thus obtaining a role that will be included in the AC, in addition to all the other information that will be normally present. In case multiple roles are needed, the `-voms` option may be used several times.

By default, all FQANs explicitly requested on the command line will be present in the returned credentials, if they were granted, and in the exact order specified, with all other FQANs following in an unspecified ordering. If a specific order is needed, it should be explicitly requested via the `-order` option. For example, the following command line:

```
voms-proxy-init --voms test:/test/Role=sgm --order /test
```

Asks for the Role sgm in the root group, and specifies that the resulting AC should begin with membership in the root group instead, while posing no requirements on the ordering of the remaining FQANs. This also means that with the above command line there is no guarantee that the role will end up as the second FQAN. If this is desired, use the following command line instead:

```
voms-proxy-init --voms test:/test/Role=sgm --order /test --order /test/Role=sgm
```

The validity of an AC created by VOMS will generally be as long as the proxy which contains it. However, this cannot always be true. For starters, the VOMS server is configured with a maximum validity for all the ACs it will create, and a request to exceed it will simply be ignored. If this happens, the output of voms-proxy-init will indicate the fact.

For example, in the following output (slightly reformatted for a shorter line then on screen):

```
[marotta@datatag6 certificates]$ voms-proxy-init --voms valerio --vomslife 50:15
Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini
Creating temporary proxy ................................... Done
Contacting  datatag6.cnaf.infn.it:50002
    [/C=IT/O=INFN/OU=Host/L=CNAF/CN=datatag6.cnaf.infn.it] "valerio" Done

Warning: datatag6.cnaf.infn.it:50002:
    The validity of this VOMS AC in your proxy is shortened to 86400 seconds!

Creating proxy ........................................... Done
Your proxy is valid until Fri Sep  8 01:55:34 2006
```

You can see that the life of the voms AC has been clearly shortened to 24 hours, even though 50 hours and 15 minutes had been requested.

If your certificate is not in the default place, you may specify it explicitly by using the –cert and –key options, like in the following example:

```
voms-proxy-init --voms valerio --cert \$HOME/cert.pem --key \$HOME/key.pem
```

Finally, in case several options have to be specified several times, profiles can be created. For examples:

```
[marotta@datatag6 marotta]$ cat voms.profile
--voms=valerio
--lifetime=50:15
--cert=/home/marotta/mycert.pem
--key=/home/marotta/mykey.pem
--order=/valerio/group1
```

followed by:

```
[marotta@datatag6 marotta]$ voms-proxy-init --conf voms.profile
```

is equivalent to:

```
[marotta@datatag6 marotta]$ voms-proxy-init --voms valerio \
             --lifetime 50:15 --cert /home/marotta/mycert.pem \
             --key /home/marotta/mykey.pem --order /valerio/group1
```

with the obvious advantages of being much less error-prone.

8.1.2 VOMS-PROXY-INFO

Once a proxy has been created, the `voms-proxy-info` command allowes the user to retrieve several information from it. The two most basic uses are:

```
[marotta@datatag6 certificates]$ voms-proxy-info
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini/CN=proxy
issuer    : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini
identity  : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini
type      : proxy
strength  : 512 bits
path      : /tmp/x509up_u502
timeleft  : 10:33:52
```

which, as you can see, prints the same information that would be printed by a plain `grid-proxy-info`, and then there is:

```
[marotta@datatag6 certificates]$ voms-proxy-info --all
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini/CN=proxy
issuer    : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini
identity  : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini
type      : proxy
strength  : 512 bits
path      : /tmp/x509up_u502
timeleft  : 11:59:59
=== VO valerio extension information ===
VO        : valerio
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini
issuer    : /C=IT/O=INFN/OU=Host/L=CNAF/CN=datatag6.cnaf.infn.it
attribute : /valerio
attribute : /valerio/asdasd
attribute : /valerio/qwerty
attribute : attributeOne = 111 (valerio)
attribute : attributeTwo = 222 (valerio)
timeleft  : 11:59:59
uri       : datatag6.cnaf.infn.it:15000
```

which prints everything that there is to know about the proxy and the included ACs. Several options enable the user to select just a subset of the information shown here.

8.1.3 VOMS-PROXY-DESTROY

The `voms-proxy-destroy` command erases an existing proxy from the system. Its basic use is:

```
[marotta@datatag6 certificates]$ voms-proxy-destroy
[marotta@datatag6 certificates]$
```

As can be seen, no output is given in case of a successful usage.

8.1.4 VOMS-PROXY-LIST

The `voms-proxy-list` command is used to interrogate a VOMS server to discover the list of group/role combination the user belongs to or may hold.

Its basic usage is:

```
[marotta@datatag6 certificates]$ voms-proxy-list --voms valerio
Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Vincenzo Ciaschini
Creating temporary proxy ............................... Done
Contacting  datatag6.cnaf.infn.it:50002
        [/C=IT/O=INFN/OU=Host/L=CNAF/CN=datatag6.cnaf.infn.it] "valerio" Done
Available attributes:
/valerio
/valerio/asdasd
/valerio/qwerty
/valerio/qwerty
```

8.1.5 VOMS-PROXY-FAKE

The `voms-proxy-fake` is used to generate a proxy containing the VOMS extension without actually having to install and setup a true VOMS server, and without having to register users into it. It is intended as a tool for testing and debugging software, and to this purpose it is also capable of generating proxies that are invalid or wrongly formatted. It can be also used to test compatibility with features of VOMS without the need to have that specific version of VOMS installed.

Its basic usage is:

```
/home/marotta/installs/testsuite/bin/voms-proxy-fake --hostcert
  myhostcert.pem --hostkey myhostkey.pem --voms prova --fqan
  /prova --uri testbed002.cnaf.infn.it:9898
```

This generates a proxy from the VO `prova`, containing the FQAN `/prova`, and with a server uri `testbed002.cnaf.infn.i` signed by the certificate corresponding to the `myhostcert.pem`, `myhostkey.pem`, and with holder the current user certificate.

The three options shown here, `-voms`, `-fqan`, `-uri` are the minimum necessary to have a properly formed VOMS proxy. Note also that if a real VOMS server's certificate and key are used in place of the test one above, then the proxy created is indistinguishable by one created by the real VOMS server, so you should always use a specially generated hostcert/hostkey pair and not rely on an existing server's one, unless you have a very good reason to do so.

Also, note that since this is a tool intended for testing and debugging, there is no DB involved, meaning that everything can be specified as an argument to –fqan, and you will find that in the proxy certificate generated. For example, if `-fqan /atlas/Role=sgm` is specified, than the `/atlas/Role=sgm` FQAN will be included in the proxy. This does not constitute a risk as long as a different host certificate than those used by the real ATLAS VO is used.

Part IV

User's Guide

9 VOMS-PROXY-INIT

This command is used to contact the VOMS server and retrieve an AC containing user attributes that will be included in the proxy certificates.

Before running voms-proxy-init, or any other client command, it is necessary to install the VO config RPM that should be distributed by the VOMS servers themselves, and has probably already been done by the system manager.

This usually results in installing files in the "/etc/grid-security/vomsdir" and "/etc/vomses" directories.

One step that cannot be done by the system manager is instead to install the user certificate. To do so, the certificate must be installed in `$HOME/.globus` either as a PKCS12 container called `usercert.p12` or `usercerd.p12`, or as a PEM certificate/key couple, called respectively `usercert.pem` and `userkey.pem`.

Permissions should be set at 600 or 400 for `usercert.pem` and 400 for `userkey.pem` and `usercert.p12`.

The voms-proxy-init command can be invoked with the following options:

–help
–usage
: These two options are synonymous and will print a short usage reminder and then quit.

–voms
: Specifies which server to contact. The parameter has the following syntax: <alias>[:<command>] where <alias> is the alias of the server as specified in the vomses files. If the same alias is associated to more than a single server, than those servers are considered replicas of each other, and are contacted in random order until one succeeds or all fail.
The [:<command>] part is optional. If not specified then the information returned will include only group membership, while if you specify :/<groupname>/Role=<rolename> then you will also get the role you asked for, provided that the server is already prepared to grant it to you.
This option can be specified multiple times, and the operations will be carried out in the exact order in which these options are specified in the command line.
You may also specify :all, which will get you all possible roles.

–version
: Prints version information and exits.

–q
–quiet
: Prints only minimal informations. *WARNING*: some vital warnings may get overlooked by this option.

–verify
: Verifies the certificate from which to create the proxy. This is not normally done, since in any case, an invalid user certificate will be detected when the proxy is actually used.

–pwstdin	Specifies that the private key's passphrase should be received from stdin instead than directly from the console.
–limited	Creates a limited certificate.
–valid	Specifies the length of the validity of both the generated proxy and the AC, measure in hours:minutes. This option should be used in preference to `-hours` or `-vomslife`, which instead specify the duration for the separate parts.
–hours	Specifies the length of the validity of the generated proxy, measure in hours. The default value is 12 hours.
–bits	Specifies the length in bits of the private key of the newly generated proxy certificate. The default value is 1024. Acceptable values are 512, 1024, 2048.
–cert	Specifies a non-standard location of the user's certificate. The default value is "$X509_USER_CERT" or, if this value is unset, "/$HOME/.globus/usercert.pem", or "/$HOME/.globus/usercert.p12" or "/$HOME/.globus/usercred.p12", in order.
–key	Specifies a non-standard location of the user's private key. The default value is "$X509_USER_KEY" or, if this value is unset, "$HOME/.globus/userkey.pem". This option is ignored if the PKCS12 format for the credentials is used.
–certdir	Specifies a non-standard location of the trusted cert (CA) directory. The default value is "/etc/grid-security/certificates".
–out	Specifies a non-standard location of the generated proxy certificate. The default value is "$X509_USER_PROXY" or, if this is empty, "/tmp/x509up_u<id>" where <id> is the user's UID.

–order	This option specifies the order in which the attributes granted by the VOMS servers should be returned. The format of the parameter for this option is: <group[:role]>, where "group" is a group name and "role" is an (optional) role name. This option may be specified multiple times, to create an ordered list of attributes. Each server will receive this list, and will strive to return the attributes he will grant in the exact order specified by this list. All attributes not on this list will be returned in an unspecified order, but after the recognized attributes. Also, should this list include an attribute unknown to a specific server, such an attribute will be simply ignored. Finally, should a server be unable to grant the first attribute of the list, it will return a warning to the user. However, this warning will only be significant for the first server contacted.
–target	This option take advantage of the capability ACs have to target themselves to a specific set of receivers, so that only those receivers should, in conforming implementation, act on the data they get, while all others should reject it. This options lets you specify a set of FQHNs, each on a separate option, that will constitute the set of targets for the generated AC.
–vomslife	This option lets you specify the validity, in h:m format, that you wish for the generated ACs. Remember that this value has only an advisory role. VOMS servers may lower this duration if the requested value exceeds the maximum they have been configured to grant. The default value of this option is "the value of the –hours or –valid options."
–globus	The version of Globus installed on the server's host. Use 20 for Globus 2.0 or Globus 2.1, 22 for Globus 2.2 and Globus 2.4, 30 for globus 3.x and 40 for globus 4.0 The default value is the value of the $GLOBUS_VERSION variable, or 22 if that is unset.
–noregen	For its normal workings, voms-proxy-init first creates a proxy with which to contact the VOMS servers, and then creates a new proxy to hold all of the returned ACs. This option skips the creation of the first proxy, and assumes that such a proxy already exists.
–separate	This option save the ACs in a separate file, instead than including them into a proxy certificate.

–ignorewarn	Specify this if you do not want to allow warnings to be printed.
–failonwarn	Specify this if you want warnings to be upgraded into errors.
–confile	Deprecated. Use –vomses instead.
–userconf	Deprecated. Use –vomses instead.
–vomses	This option specifies the location of an additional directory in which to search for vomses files or directly such a file. `/etc/vomses` and `$HOME/.voms/vomses` are always added by default. This option may be specified multiple times.
–conf	Lets you specify a file from which options will be loaded. This file should have exactly one option per line, and option that do have values should be specified in the format "option=value".
–debug	This option prints a series of additional debug informations on stdout. The additional output returned by this option should *always* be included into bug reports for the voms-proxy-init command. User should not, however, ever rely on informations printed by this options. Both content and format are guaranteed to change between software releases.
–policy	This options allows a policy file to be included in the proxy.
–pl	
–policy-language	This option specifies th OID for the string policy language.
–proxyver	This option specifies to which version of the proxy certificate format will the generated format conform to. Possible values are 2, 3 and 4, while the default is the one implied by the –globus option.
–rfc	This is a synonymous of `-proxyver 4`.
–old	This is a synonym for –proxyver 2.
–path-length	This option specifies the maximum path length for the proxy. Note that this is NOT the certificate path length, but the policy path length.
–include	This options specifies the location of a file that shouls be include "as-is" inside the proxy.
–list	This option turns the command in `voms-proxy-list`. When this happens, several other options get ignored. See that command manual for a description of the still valid options and their meaning.
–timeout	This options allows to specify the maximum number of seconds that voms-proxy-init will wait on an hanged connection to the server. The default is 60 seconds.

–includeac	This option specifies a file containing an AC that will be included in the proxy.

10 VOMS-PROXY-INFO

This command is used to print to the screen the informations included in an already generated VOMS proxy.

The configuration is the same as voms-proxy-init.

The following options may be used:

–help	
–usage	These two options are synonymous and will print a short usage reminder and then quit.
–debug	This option prints a series of additional debug informations on stdout. The additional output returned by this option should *always* be included into bug reports for the voms-proxy-info command. User should not, however, ever rely on informations printed by this options. Both content and format are guaranteed to change between software releases.
–version	Prints version information and exits.
–conf	Lets you specify a file from which options will be loaded. This file should have exactly one option per line, and option that do have values should be specified in the format "option=value".
–file	This option lets you specify a non-standard location of the user proxy. The default value is "$X509_USER_PROXY" or, if this is empty, "/tmp/x509up_u<id>", where <id> is the user's UID.
–subject	Prints the DN of the proxy's subject.
–issuer	Prints the DN of the proxy-s issuer.
–identity	Prints the DN corresponding to the identity represented by the proxy.
–type	Prints the proxy's type.
–strength	Prints the length (in bits) of the private key.
–valid	Prints the start and end validity times.
–time	Prints the end validity as a number of seconds for which the object will still be valid.
–timeleft	Prints how much time is left to the proxy until expiration.
–vo	Prints the VO name of each AC.
–text	Prints all of the certificate.
–info	Lets "–subject", "–issuer", "–valid" and "–time" also apply to ACs, and prints attributes values.

–extra	Prints extra informations that were included in the proxy.
–path	Prints the path at which the proxy was found.
–all	Prints everything. (Implies all other options.)
–fqan	Specifies that attributes should be printed in the FQAN format. (default)
–extended	Specifies that attributes should be printed in the extended format.
–exists	Activates the "–hours" and "–bits" options.
–hours	Verifies that the proxy, and the ACs if "–info" was specified, will be valid for at least <H> hours.
–bits	Verifies that the proxy key has at least <B> bits.
–acexists	Verifies wether an AC of the specified VO is included in the proxy.
–acsubject	Prints the DN of the AC's subject.
–acissuer	Prints the DN of the AC's issuer.
–actimeleft	Prints how much time is left to the AC before expiration.
–serial	Prints the serial number of the AC.
–targets	Prints the targets of the AC.
–included-file	Prints the contents of the file include with the **–include** option of voms-proxy-init.
–uri	Prints the URI of the issuing server.
–chain	Prints informations about the whole chain of certificates included in the proxy, not just the last one.
–dont-verify-ac	Skips AC verification. The same effect as this option may be obtained by specifying the `VOMS_PROXY_INFO_DONT_VERIFY_AC` environment variable.

11 VOMS-PROXY-DESTROY

This command destroys an already existing VOMS proxy.

No configuration is needed.

The following options may be used:

–debug	This option prints a series of additional debug informations on stdout. The additional output returned by this option should *always* be included into bug reports for the voms-proxy-info command. User should not, however, ever rely on informations printed by this options. Both content and format are guaranteed to change between software releases.
–version	Prints version information and exits.

–conf	Lets you specify a file from which options will be loaded. This file should have exactly one option per line, and option that do have values should be specified in the format "option=value".
–quiet	Prints only minimal informations. *WARNING*: some vital warnings may get overlooked by this option.
–file	This option lets you specify a non-standard location of the user proxy. The default value is "$X509_USER_PROXY" or, if this is empty, "/tmp/x509up_u<id>", where <id> is the user's UID.
–dryrun	Only prints messages, but do not take any actions.

12 VOMS-PROXY-LIST

This command allows a user to get a list of all the group/role conbination that he may request.

The following options may be used:

–help	
–usage	These two options are synonymous and will print a short usage reminder and then quit.
–voms	Specifies which server to contact. The parameter has the following syntax: <alias>[:<command>] where <alias> is the alias of the server as specified in the vomses files. If the same alias is associated to more than a single server, than those servers are considered replicas of each other, and are contacted in random order until one succeeds or all fail. The [:<command>] part is optional. If not specified then the information returned will include only group membership, while if you specify :/<groupname>/Role=<rolename> then you will also get the role you asked for, provided that the server is already prepared to grant it to you. This option can be specified multiple times, and the operations will be carried out in the exact order in which these options are specified in the command line. You may also specify :all, which will get you all possible roles.
–version	Prints version information and exits.
–quiet	Prints only minimal informations. *WARNING*: some vital warnings may get overlooked by this option.

–pwstdin	Specifies that the private key's passphrase should be received from stdin instead than directly from the console.
–cert	Specifies a non-standard location of the user's certificate. The default value is "$X509_USER_CERT" or, if this value is unset, "/$HOME/.globus/usercert.pem", or "/$HOME/.globus/usercert.p12" if the former one is not present..
–key	Specifies a non-standard location of the user's private key. The default value is "$X509_USER_KEY" or, if this value is unset, "$HOME/.globus/userkey.pem". This option is ignored if the PKCS12 format for the credentials is used.
–certdir	Specifies a non-standard location of the trusted cert (CA) directory. The default value is "/etc/grid-security/certificates".
–out	Specifies a non-standard location of the generated proxy certificate. The default value is "$X509_USER_PROXY" or, if this is empty, "/tmp/x509up_u<id>" where <id> is the user's UID.
–globus	The version of Globus installed on the server's host. Use 20 for Globus 2.0 or Globus 2.1, 22 for Globus 2.2 and Globus 2.4, 30 for globus 3.x and 40 for globus 4.0 The default value is the value of the $GLOBUS_VERSION variable, or 22 if that is unset.
–noregen	For its normal workings, voms-proxy-init first creates a proxy with which to contact the VOMS servers, and then creates a new proxy to hold all of the returned ACs. This option skips the creation of the first proxy, and assumes that such a proxy already exists.
–ignorewarn	Specify this if you do not want to allow warnings to be printed.
–failonwarn	Specify this if you want warnings to be upgraded into errors.
–confile	Deprecated. Use –vomses instead.
–userconf	Deprecated. Use –vomses instead.
–vomses	This option specifies the location of an additional directory in which to search for vomses files or directly such a file. `/etc/vomses` and `$HOME/.voms/vomses` are always added by default. This option may be specified multiple times.

–conf	Lets you specify a file from which options will be loaded. This file should have exactly one option per line, and option that do have values should be specified in the format "option=value".
–debug	This option prints a series of additional debug informations on stdout. The additional output returned by this option should *always* be included into bug reports for the voms-proxy-init command. User should not, however, ever rely on informations printed by this options. Both content and format are guaranteed to change between software releases.
–list	This is always specified by default.
–timeout	This options allows to specify the maximum number of seconds that voms-proxy-init will wait on an hanged connection to the server. The default is 60 seconds.

13 VOMS-PROXY-FAKE

This command allows the creation of VOMS proxies without the need of a working server. This command is intended as a tool for testing and debugging. As such, it is possible to create proxies that are invalid.

The following options may be used:

–help	
–usage	These two options are synonymous and will print a short usage reminder and then quit.
–version	Prints version information and exits.
–debug	This option prints a series of additional debug informations on stdout. The additional output returned by this option should *always* be included into bug reports for the voms-proxy-init command. User should not, however, ever rely on informations printed by this options. Both content and format are guaranteed to change between software releases.
–verify	This option verifies the user certifiacte used to create the proxy.
–pwstdin	Specifies that the private key's passphrase should be received from stdin instead than directly from the console.
–limited	Creates a limited certificate.
–hours	Specifies the length of the validity of the generated proxy, measure in hours. The default value is 12 hours.

–vomslife	This option lets you specify the validity, in h:m format, that you wish for the generated ACs. Remember that this value has only an advisory role. The default value of this option is "the value of the –hours option."
–bits	Specifies the length in bits of the private key of the newly generated proxy certificate. The default value is 512. Acceptable values are 512, 1024, 2048.
–debug	This option prints a series of additional debug informations on stdout. The additional output returned by this option should *always* be included into bug reports for the voms-proxy-init command. User should not, however, ever rely on informations printed by this options. Both content and format are guaranteed to change between software releases.
–cert	Specifies a non-standard location of the user's certificate. The default value is "$X509_USER_CERT" or, if this value is unset, "/$HOME/.globus/usercert.pem", or "/$HOME/.globus/usercert.p12" or "/$HOME/.globus/usercred.p12", in order.
–key	Specifies a non-standard location of the user's private key. The default value is "$X509_USER_KEY" or, if this value is unset, "$HOME/.globus/userkey.pem". This option is ignored if the PKCS12 format for the credentials is used.
–certdir	Specifies a non-standard location of the trusted cert (CA) directory. The default value is "/etc/grid-security/certificates".
–out	Specifies a non-standard location of the generated proxy certificate. The default value is "$X509_USER_PROXY" or, if this is empty, "/tmp/x509up_u<id>" where <id> is the user's UID.
–limited	Creates a limited certificate.
–q	
–quiet	Prints only minimal informations. *WARNING*: some vital warnings may get overlooked by this option.
–conf	Lets you specify a file from which options will be loaded. This file should have exactly one option per line, and option that do have values should be specified in the format "option=value".
–voms	This option lets you specify the VO to which the AC will belong.

–target	This options lets you specify a set of FQHNs, each on a separate option, that will constitute the set of targets for the generated AC.
–globus	The version of Globus installed on the server's host. Use 20 for Globus 2.0 or Globus 2.1, 22 for Globus 2.2 and Globus 2.4, 30 for globus 3.x and 40 for globus 4.0 The default value is the value of the $GLOBUS_VERSION variable, or 22 if that is unset.
–proxyver	This option specifies to which version of the proxy certificate format will the generated format conform to. Possible values are 2, 3 and 4, while the default is the one implied by the –globus option.
–policy	This options allows a policy file to be included in the proxy.
–pl	
–policy-language	This option specifies th OID for the string policy language.
–path-length	This option specifies the maximum path length for the proxy. Note that this is NOT the certificate path length, but the policy path length.
–separate	This option save the ACs in a separate file, instead than including them into a proxy certificate.
–uri	This option lets you specify the URI to the fictional server which created this AC.
–hostcert	This option lets you specify the host certificate that will be used to sign the AC.
–hostkey	This option lets you specify the private key of the certificate that will be used to sign the AC.
–fqan	This option lets you specify the FQAN that will be included in the AC. This option may be specified multiple times, and all the FQANs will be included, in the order in which they are found on the command line.
–newformat	Creates AC in the newer, correct format. Warning! Specifying this option breaks compatibility with VOMS APIs version 1.6.8 or before.
–include	Insert a the specified file in the AC in a non-critical extension.
–newsubject	Sets the subject that the proxy will have.
–newissuer	Sets the issuer that the proxy will have.
–newserial	Sets the serial number of he proxy.
–pastac	The AC validity will start in the past for the specified time. It may be specified as seconds, HH:MM, or HH:MM:SS.
–pastproxy	The proxy validity will start in the past for the specified time. It may be specified as seconds, HH:MM, or HH:MM:SS.

–nscert	Specifies the content of the Netscape Certificate Extension. Acceptable values, separated by commas, are: client, server, email, objsign, sslCA, emailCA, objCA.
–keyusage	Specifies the content of the Key Usage extension. Acceptable values, separated by commas, are: digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly, decipherOnly.
–extkeyusage	Specifies te content of the Extended Key Usage Extensions. Acceptable values, separated by commas, are: serverAuth, clientAuth, codeSigning, emailProtection, timeStamping, msCodeInd, msCodeCom, msCTLSign, msSGC, msEFS, msSGC, deltaCRL.
–selfsigned	The certificate will be a self-signed certificate and ave CA=true in the Basic Constraints Extension.
–ga	This options allows to specify a GA to be added to the AC. The syntax is: "name=value (context)".
–extension	Adds the specified extension to the certificate. The syntax of the argument is: oid[/criticality]value. oid is the OID to use. /criticality must be either /true or /false, and specifies whether the extension is critical. If absent, /false is the default. value is the content of the extension. It is either: :text tilde hex content +filename.
–acextensions	This option is the same as –extension, except that the extension is added to the AC.
–voinfo	Specifies the name of a file which contains options for several VOs. It is divided into section, each section starting with [vo] and includeing the options and their values for that VO, one option per line.

Note that to create a proxy that will be at least correctly formatted, the following options are necessary: `-hostcert`, `-hostkey`, `-voms`, `-uri` and `-fqan`.

Also, as a temporary limitation, only one AC may be included in the proxy. This limitation will be removed in future versions.

Part V

Application Programming Interfaces

The VOMS API already come with their own documentation in doxygen format. However, that documentation is little more than a simple enumeration of functions, with a very terse description.

The aim of this document is different. Here the intention is not only to describe the different functions that comprise the API, but also to show how they are supposed to work together, what particular care the user needs to take when calling them, what should be done to mantain compatibility between the different versions, etc. . .

Compatibility

Finally, please note that everything not explicitly defined in this section shall be considered a private detail and subject to change without notice.

14 C++ API

There are three basic classes: `data`, `voms` and `vomsdata`.

14.1 The data structure

The first one, `data` contains the data regarding a single attribute, giving its specification in terms of Groups, Roles and Capabilities. It is defined as follows:

```
struct data {
  std::string group;
  std::string role;
  std::string cap;
};
```

All the values of these strings must be composed from regular expression: `[a-ZA-Z0-9_˙/]*`.

14.1.1 GROUP

This field contains the name of a group which the user belongs into. The format of entries in this group is reminiscent of the structure of pathnames, and is the following:

/group/group/.../group

where the name of the first group is by convention the name of the Virtual Organization (VO), while each other */group* component is a subgroup of the group immediately preceding it on the left. The character '/' is therefore acceptable as part of a group name.

This field MUST always be filled.

14.1.2 ROLE

This field contains the name of the role which the user owns in the group specified by `group`. If the user does not own any particular role in that group, than this field contains the value "NULL".

Compatibility

Or it may be set to `NULL`.

14.1.3 CAP

This field details a capability that the user has as a member of the group specified by `group` while owning the role specified by `role`. If there is no specific capability, than this value is "NULL".

Compatibility

Or it may be set to `NULL`.

No specific format is associated to a capability. They are basically free-form strings, whose value should be agreed between the AA and the Attribute verifier.

They are deprecated. No application should ever rely on them.

14.2 THE VOMS STRUCTURE

The second one, `voms` is used to group together all the informations that can be gleaned from a single AC, and is defined as follows:

```
enum data_type {
  TYPE_NODATA,  /*!< no data */
  TYPE_STD,     /*!< group, role, capability triplet */
  TYPE_CUSTOM   /*!< result of an S command */
};

struct voms {
  /* Private data and methods omitted. */
  friend class vomsdata;
  int version;
  int siglen;
  std::string signature;
  std::string user;
  std::string userca;
  std::string server;
  std::string serverca;
  std::string voname;
  std::string uri;
  std::string date1;
  std::string date2;
  data_type type;
  std::vector<data> std;
  std::string custom;
  /* Data below this line only makes sense if version >= 1 */
  std::vector<std::string> fqan;
```

```
  std::string serial;
  /* Data below this line is private. */
  voms(const voms &);
  voms();
  voms &operator=(const voms &);
  ~voms();
  AC *GetAC();
  std::vector<attributelist>& GetAttributes();
  std::vector<std::string> GetTargets();
};
```

The purpose of this structure is to present, in a readable format, the data that has been included in a single Attribute Certificate (AC). While the various public fields may be freely modified to simplify internal coding, such changes have no effect on the underlying AC. Let's examine the various fields in detail.

14.2.1 VERSION

```
int version;
```

This field specifies the version of this structure that is currently being used. A value of 0 indicates that it comes from an old format extension, while a value of 1 indicates that this structure comes from an AC.

Compatibility

Starting from VOMS 1.6.0, support for version 0 is no longer present. What this means is that now VOMS 1.6.0 or greater is incompatible with VOMS 1.1.x or lesser.

14.2.2 SIGLEN

```
int siglen;
```

The length of the data signature.

14.2.3 SIGNATURE

```
std::string signature;
```

This field contains a copy of the signature of the AC.

14.2.4 USER

```
std::string user;
```

This field contains the subject of the holder's certificate in slash-separated format.

14.2.5 USERCA

```
std::string userca;
```

This field contains the subject of the CA that issued the holder's certificate, in slash-separated format.

14.2.6 SERVER

```
std::string server;
```

This field contains the subject of the certificate that the AA used to issue the AC, in slash-separated format.

14.2.7 SERVERCA

```
std::string serverca;
```

This field contains, in slash-separated format, the subject of the CA that issued the certificate that the AA used to issue the AC.

14.2.8 VONAME

```
std::string voname;
```

This field contains the name of the Virtual Organization (VO) to which the rest of the data contained in this structure applies to.

14.2.9 URI

```
std::string uri;
```

This is the URI at which the AA that issued this particular AC can be contacted. Its format is:

fqdn:*port*

where *fqdn* is the Fully Qualified Domain Name of the server which hosts the AA, and *port* is the port at which the AA can be contacted on that server.

14.2.10 DATE1, DATE2

```
std::string date1;
std::string date2;
```

These are the dates of start and end of validity of the rest of the informations. They are in a string representation readable to humans, but they may be easily converted back to their original format.

Here follows a code example doing that conversion:

```
ASN1_TIME *
convtime(std::string data)
{
  ASN1_TIME *t= ASN1_TIME_new();

  t->data   = (unsigned char *)(data.data());
  t->length = data.size();
  switch(t->length) {
```

```
  case 15:
    t->type = V_ASN1_GENERALIZEDTIME;
    break;
  default:
    ASN1_TIME_free(t);
    return NULL;
  }
  return t;
}
```

14.2.11 TYPE

```
data_type type;
```

This datum specifies the type of data that follows. It can assume the following values:

TYPE_NODATA OBSOLETE

TYPE_CUSTOM OBSOLETE

Compatibility

Both TYPE_NODATA and TYPE_CUSTOM were only present in version 0. Since that version is no longer supported, their content is undocumented, and MUST not be used by applications.

TYPE_STD The data will contain (group, role, capabilities) triples.

14.2.12 STD

```
std::vector<data> std;
```

This vector contains all the attributes found in an AC, in the exact same order as they were found, in the format specified by the `data` structure. It is only filled if the value of the `type` field is `TYPE_STD`.

14.2.13 CUSTOM

```
std::string custom;
```

This field contains the data returned by the "S" server command, and it is only filled if the `type` value id `TYPE_CUSTOM`.

14.2.14 FQAN

```
std::vector<std::string> fqan;
```

This field contains the same data as the `std` field, but specified in the Fully Qualified Attribute Name (FQAN) format.

14.2.15 SERIAL

```
std::string serial;
```

This field contains a text representation of the serial number of the certificate.

14.2.16 GETAC()

```
AC *GetAC();
```

This method returns a copy of the underlying AC. This means that eventual changes are of no consequence.

14.2.17 GETATTRIBUTES()

```
std::vector<attributelist>& GetAttributes();
```

This method is used to retrieve the list of generic attributes that were present in the AC. The result is a vector of `attributelist` structures, one each for every source of attributes. The definition of `attributelist` follows;

```
struct attributelist {
  std::string grantor;
  std::vector<attribute> attributes;
};
```

here `grantor` is a string identifying who granted the attributes, while `attributes` is a vector of the attributes themselves, defined as follows:

```
struct attribute {
  std::string name;
  std::string qualifier;
  std::string value;
};
```

In this structure, `name` is the name of the attribute, `value` is its value, while `qualifier` is an optional qualifier. `qualifier` may be the empty string if no special qualifier was specified for this attribute.

14.2.18 GETTARGETS()

```
std::vector<std::string> GetTargets();
```

This method is used to retrieve the list of hosts this AC is targeted to. If there is no target, than the vector is empty, otherwise it contains all the targets of this AC.

14.2.19 VOMS::VOMS()

```
voms::voms()
```

This is the standard default constructor. Please note that a structure created this way would not contain any real data. The only use for this constructor is to create a "placeholder" structure to which you will copy data using the copy operator.

14.2.20 VOMS::VOMS(CONST VOMS &)

```
voms::voms(const voms \&)
```

This is the standard copy constructor. Structures allocated via this method will retain an exact copy of the data of their source. Use of either this constructor or the assignment operator are the *only* supported method to copy a `voms` object.

14.2.21 VOMS::OPERATOR=(CONST VOMS &)

```
voms::voms& operator=(const voms \&)
```

This defines an assignment operator between two different `voms` structures. Use of either this constructor or the copy constructor are the *only* supported method to copy a `voms` object.

14.3 VOMSDATA

The purpose of this object is to collect in a single place all informations present in a VOMS extension. It is defined so.

```
struct vomsdata {
  /* Private data and methods omitted. */

public:
  verror_type error; /*!< Error code */

  vomsdata(std::string voms_dir = "",
           std::string cert_dir = "");
  bool LoadSystemContacts(std::string dir = "");
  bool LoadUserContacts(std::string dir = "");
  std::vector<contactdata> FindByAlias(std::string alias);
  std::vector<contactdata> FindByVO(std::string vo);
  void Order(std::string att);
  void ResetOrder(void);
  void AddTarget(std::string target);
  std::vector<std::string> ListTargets(void);
  void ResetTargets(void);
  std::string ServerErrors(void);
  bool Retrieve(X509 *cert, STACK_OF(X509) *chain,
                recurse_type how = RECURSE_CHAIN);
  bool Contact(std::string hostname, int port,
               std::string servsubject,
               std::string command);
  bool ContactRaw(std::string hostname, int port,
                  std::string servsubject,
                  std::string command,
                  std::string &raw, int &version);
  void SetVerificationType(verify_type how);
  void SetLifetime(int lifetime);
  bool Import(std::string buffer);
  bool Export(std::string &data);
  bool DefaultData(voms &);
  std::vector<voms> data;
  std::string workvo;
  std::string extra_data;
```

```
  std::string ErrorMessage(void);
  bool RetrieveFromCtx(gss_ctx_id_t context, recurse_type how);
  bool RetrieveFromCred(gss_cred_id_t credential, recurse_type how);
  bool Retrieve(X509_EXTENSION *ext);
  bool RetrieveFromProxy(recurse_type how);
  ~vomsdata();
  vomsdata(const vomsdata &);
  bool Retrieve(FILE* file, recurse_type how);
  void SetRetryCount(int retrycount);
  void SetVerificationTime(time_t when);
  bool LoadCredentials(X509 *cert, EVP_PKEY *key, STACK_OF(X509) *
     chain);
  bool ContactRESTRaw(const std::string\& hostname, int port,
                      const std::string\& command,
                      std::string raw, int version, int timeout);
};
```

Let us see the fields in detail.

14.3.1 ERROR

```
verror_type error;
```

This field contains the error code returned by one of the methods. Please note that the value of this field is only significant if the *last* method called returns an error value. Also, the value of this field is subject to change without notice during method executions, regardless of whether an error effectively occurred.

The possible values returned are the following:

```
enum verror_type {
  VERR_NONE,
  VERR_NOSOCKET,
  VERR_NOIDENT,
  VERR_COMM,
  VERR_PARAM,
  VERR_NOEXT,
  VERR_NOINIT,
  VERR_TIME,
  VERR_IDCHECK,
  VERR_EXTRAINFO,
  VERR_FORMAT,
  VERR_NODATA,
  VERR_PARSE,
  VERR_DIR,
  VERR_SIGN,
  VERR_SERVER,
  VERR_MEM,
```

```
  VERR_VERIFY,
  VERR_TYPE,
  VERR_ORDER,
  VERR_SERVERCODE,
  VERR_NOTAVAIL
};
```

In general, a first idea of what each code means can be gleaned from the code name, but every method description will document what errors its execution may generate and on which conditions.

14.3.2 DATA

This field contains a vector of `voms` structures, in the exact same order as the corresponding ACs appeared in the proxy certificate, and containing the informations present in that AC.

14.3.3 VOMSDATA::VOMSDATA(STD::STRING VOMS_DIR, STD::STRING CERT_DIR)

```
vomsdata::vomsdata(std::string voms_dir='''', std::string cert_dir=''
   '')}
```

This is the standard constructor that also doubles as the default constructor.
voms_dir This is the directory where the VOMS server's certificates are kept. If this value is empty (the default), then the value of `$X509_VOMS_DIR` is considered, and if this is also empty than its default is `/etc/grid-security/vomsdir`.
cert_dir This is the directory where the CA certificates are kept. If this value is empty (the default), then the value of `$X509_CERT_DIR` is considered, and if this is also empty than its default is `/etc/grid-security/certifica`

Compatibility

> This function is the only supported way to create and initialize a `vomsdata` structure other than the copy constructor. It is forbidden to ever take the `sizeof()` of this class.

The default values are strongly suggested. If you want to hardcode specific ones, think very thouroghly about the loss of configurability that it would entail.

14.3.4 BOOL VOMSDATA::LOADSYSTEMCONTACTS(STD::STRING DIR)

bool `vomsdata::LoadSystemContacts(std::string dir = '''')}`

This function loads the vomses files that are shared system-wide.
dir This is the directory in which the various vomses files are kept. If left blank, it defaults to `/etc/vomses`.

RETURNS
The return value is true if all went well and false otherwise. In the latter case the `vomsdata::error` member becomes significant, and it may assume the following values:

VERR_DIR	The function tried to access something that either was not a directory or a regular file, could not be read, or it had the wrong permissions. The acceptable permissions are, at the most, 644 for files and 755 for directories.
VERR_FORMAT	The file was not in the expected format.

14.3.5 BOOL VOMSDATA::LOADUSERCONTACTS(STD::STRING DIR)

```
bool vomsdata::LoadUserContacts(std::string dir = '''');
```

This function loads the vomses files that are user-specific.
dir This is the directory in which the various vomses files are kept. If left blank, it defaults to `$VOMS_USERCONF`. If this is also empty, then the last default is `/.voms/vomses`.

RETURNS
The return value is true if all went well and false otherwise. In the latter case the `vomsdata::error` member becomes significant, and it may assume the following values:

VERR_DIR	The function tried to access something that either was not a directory or a regular file, could not be read, or it had the wrong permissions. The correct permissions are at the most 644 for files and 755 for directories.
VERR_FORMAT	The file was not in the expected format.

14.3.6 STD::VECTOR<CONTACTDATA> VOMSDATA::FINDBYALIAS(STD::STRING ALIAS)

```
std::vector$<$contactdata$>$ vomsdata::FindByAlias(std::string alias)
   ;
```

Where `contactdata` is defined as:

```
struct contactdata {   /*!< You must never allocate directly this
   structure.
                                Its sizeof() is subject to change
                                   without notice.
                                            The only supported way
                                               to obtain it is via
                                               the
                                FindBy* functions. */
  std::string nick;    /*!< The alias of the server */
  std::string host;    /*!< The hostname of the server */
  std::string contact; /*!< The subject of the server's certificate
     */
  std::string vo;      /*!< The VO served by this server */
  int         port;    /*!< The port on which the server is listening
      */
};
```

This function looks in the vomses files loaded by `vomsdata::LoadSystemContacts()` and `vomsdata::LoadUserContacts()` for servers that have been registered with a particular alias.
alias The alias that will be searched for. The search will be case sensitive.

RETURNS
The return value is a vector containing the data (in `contactdata` format) of all the servers known by the system that go by the specified alias. This function does not have an error code, but the vector may be empty if no servers satisfying the query are found or if there are no known servers altogether, typically because the Load*Contacts() function have not been called.

14.3.7 VOID VOMSDATA::ORDER(STD::STRING ATTRIBUTE)

```
void vomsdata::Order(std::string attribute);
```

This function should be called before the various Contact*() ones, and it is used to specify in which order the clients would like to have the attributes returned by the server.

It can be called multiple times, each time specifying a new attribute, creating in this way an ordered list of attributes. Then, when the server is contacted, it will examine this list of attributes against the one it would grant the client, and order the latter in the same way, with the following provisions:

- All attributes not explicitly indicated in the order list will be placed in an unspecified order after all the specified ones.
- An attribute present in the order list but not present among the attributes that the server is prepared to grant will be silently ignored.

attribute The attribute that should be ordered

Compatibility

Both FQAN (e.g: /group/Role=role) and legacy (e.g: <group name>:<role name>) formats are supported. However, please note that older versionsmay not be able to handle the FQAN format.

SEE ALSO
ResetOrder

14.3.8 VOID VOMSDATA::RESETORDER(VOID)

```
void vomsdata::ResetOrder(void);
```

This function clears the list of attributes that has been setup via calls to the Order() function. **SEE ALSO**
Order

14.3.9 VOID VOMSDATA::ADDTARGET(STD::STRING TARGET)

```
void vomsdata::AddTarget(std::string target);
```

This function takes advantage of ACs capability to target themselves to a specific set of hosts. Through consecutive calls of this function, the user can target the AC that the server will generate to any set of hosts it likes. Obviously, this function should be called before the Contact*() ones.
target The name of the host to which the AC will be targeted. The name MUST be expressed in Fully Qualified Host Name format. **SEE ALSO**
ListTargets, ResetTargets

14.3.10 STD::VECTOR<STD::STRING> VOMSDATA::LISTTARGETS(VOID)

```
std::vector$<$std::string$>$ vomsdata::listTargets(void);
```

function returns a vector containing the list of hosts that will constitute the targets that will be include in the AC.

RETURNS
A vector whose members are the FQHNs of the machines against which the AC will be targeted. This may be empty if the list has been cleared or it has never been filled.

SEE ALSO
AddTarget, ResetTargets

14.3.11 VOID VOMSDATA::RESETTARGETS(VOID)

```
void vomsdata::ResetTargets(void);
```

This function clears the list of targets for an AC. **SEE ALSO**
AddTarget, ListTargets

14.3.12 STD::STRING VOMSDATA::SERVERERRORS()

```
std::string vomsdata:ServerErrors();
```

In case one of the other functions returned a `VERR_SERVER` message, meaning that some error has occurred on the server side of a connection, calling this function MAY return a message from the server itself detailing the error.

RETURNS
The error message itself

14.3.13 VOID VOMSDATA::SETVERIFICATIONTYPE(VERIFY_TYPE HOW)

```
void vomsdata::SetVerificationType(verify\_type how);
```

This function sets the type of AC verification done by the Retrieve() and Contact() functions. The choices are detailed in the verify_type type.

```
enum verify_type {
  VERIFY_FULL       = 0xffffffff,
  VERIFY_NONE       = 0x00000000,
  VERIFY_DATE       = 0x00000001,
  VERIFY_TARGET     = 0x00000002,
  VERIFY_KEY        = 0x00000004,
  VERIFY_SIGN       = 0x00000008,
  VERIFY_ORDER      = 0x00000010,
  VERIFY_ID         = 0x00000020,
```

```
  VERIFY_CERTLIST  = 0x00000040
};
```

The meaning of these types is the following:

VERIFY_DATE This flag verifies that the current date is within the limits specified by the AC itself.

VERIFY_TARGET This flag verifies that the AC is being evaluated in a machine that is included in the target extension of the AC itself.

VERIFY_KEY This flag is for a future extension and is unused at the moment.

VERIFY_SIGN This flag verifies that the signature of the AC is correct.

VERIFY_ORDER This flag verifies that the attributes present in the AC are in the exact order that was requested. Please note that this can ONLY be done when examining an AC right after generation with the Contact() function. This flag is meaningless in all other cases.

VERIFY_ID This flag verifies that the holder information present in the AC is consistent with:

1. The enveloping user proxy in case the AC was contained in one.
2. The user's own certificate in case the AC was received without an enclosing proxy.

VERIFY_CERTLIST This flag verifies the issuer certificates included with the AC, if present.

VERIFY_FULL This flag implies all other verifications.

VERIFY_NONE This flag disables all verifications.

These flags can be combined by OR-ing them together. However, if VERIFY_NONE is OR-ed to any other flag, it can be dismissed, while if VERIFY_FULL is OR-ed to any other flag, all other flags ca be dismissed.

If this function is not explicitly called by the user, a VERIFY_FULL flag is considered to be in effect.

14.3.14 VOID VOMSDATA::SETLIFETIME(INT LIFETIME)

```
void vomsdata::SetLifetime(int lifetime);
```

This function should be called before the Contact*() ones. Its aim is to set the requested lifetime for the AC that the server would create. Please note that this is only a suggestion, and that the server may well override it if the requested time is against its own policy.

lifetime The requested lifetime, in seconds.

14.3.15 BOOL LOADCREDENTIALS(X509 *CERT, EVP_PKEY *KEY, STACK_OF(X509) *CHAIN)

```
bool LoadCredentials(X509 *cert, EVP_PKEY *key, STACK_OF(X509)
*chain);
```

This function loads the credentials from the specified locations to use them for connecting to the voms server.

cert The certificate to load.

key The private key associate to the certificate.
chain The certificate chain associated to the certificate.

The result value is a boolean that is `true` if and only if there have not been errors. If the value is `false`, then the operation failed because of memory problems.

14.3.16 VOID VOMSDATA::SETVERIFICATIONTIME(TIME_T WHEN)

```
void vomsdata::SetVerificationTime(time_t when);
```

This function modifies the behaviour of the AC verification. The verification is done as if the time, rather than being the current time, was the time passed at this function. This is useful to verify, for example, if an AC will be valid 12 hours from now. Note that obviously this cannot predict certificate revocation, so the CRL used are still the ones available at the time when the verification is done.

Passing '0' as an argument disables this capability, and restores the current time as the time of verification.
when The time that will be considered 'current' for verification purposes.

14.3.17 VOID VOMSDATA::SETRETRYCOUNT(INT COUNT)

```
void vomsdata::SetRetryCount(int count)
```

This option allows you to specify how many attempts should be done to contact a server before giving up. The default is 1.
count How many times to retry contacting a VOMS server.

14.3.18 BOOL VOMSDATA::RETRIEVE*()

```
bool vomsdata::Retrieve(X509 *cert, STACK_OF(X509) *chain,
   recurse_type how = RECURSE_CHAIN);
bool RetrieveFromCtx(gss_ctx_id_t context, recurse_type how);
bool RetrieveFromCred(gss_cred_id_t credential, recurse_type how);
bool Retrieve(X509_EXTENSION *ext);
bool RetrieveFromProxy(recurse_type how);
bool Retrieve(FILE &file, recurse_type how);
```

This set of functions retrieve a VOMS AC from a VOMS-enabled credential, executes the verifications requested by the `SetVerificationType()` function and interprets the data.

Each function retrieves the AC from their corresponding credential type. `RetrieveFromProxy()` retrieves the credential from an existing proxy.
cert This is the X509 proxy certificate from which we want to retrieve the informations.
chain This is the certificate chain associated to the proxy certificate. This parameter is only significant if the value of the next parameter is `RECURSE_CHAIN`.
context A GSS context. This is created in a server when a GSS authenticated connection occurs and includes the other side's credentials.
credential A GSS credential.
file A file which contains the proxy to parse.
ext An X509 extension, which should be the extension used by VOMS to include the AC.
how This parameters may have two values:

RECURSE_NONE meaning that the VOMS extension MUST be found in the certificate proper, or

RECURSE_CHAIN meaning that if the VOMS extension are not found in the certificate proper, the certificate chain may be descended until either the extension is found or the chain ends.

The default value is `RECURSE_CHAIN`.

`RECURSE_NONE` should only be used in special circumstances, since it is guaranteed that in a normal Grid environment the process of credential delegation will make the VOMS extension to be only present in the certificate chain.

The result value is a boolean that is `true` if and only if there have not been errors. If the value is `false`, then you should check the error code, which may have one of the following values:

VERR_PARAM	There was something wrong with the parameters passes to the function, or some of the required information (holder, etc...) is empty.
VERR_FORMAT	If the format of the data is unknown (e.g. neither an AC nor an old-style blob.
VERR_NOIDENT	If it was impossible to discover the holder of the AC. This may also be returned by one of the `RetrieveFrom*()` methods if the corresponding credential could not be found.
VERR_NOINIT	The vomsdata object hasn't been properly initialized. Most likely the voms_dir and ca_dir parameters are empty.
VERR_PARSE	There has been some problem in parsing the AC or blob.
VERR_VERIFY	It was not possible to verify the signature.
VERR_SERVER	It was not possible to properly identify the Attribute Issuer.
VERR_TIME	The check on the validity dates failed.
VERR_IDCHECK	The holder of the AC is not the same entity as the holder of the enclosing certificate.
VERR_NOTAVAIL	This may be returned from `RetrieveFromCred()`, `RetrieveFromCtx` or `RetrieveFromProxy()` if the version of the library linked is the one withouth Globus dependencies.

SEE ALSO
SetVerificationType()

14.3.19 BOOL VOMSDATA::CONTACT(STD::STRING HOSTNAME, INT PORT, STD::STRING SERVSUBJECT, STD::STRING COMMAND)

This function is used to contact a specified server and use the received AC to fill the vomsdata structure.
hostname The fully qualified hostname of the machine on which the server runs.
port The port number on which the server is listening.
servsubject The subject of the server' certificate.
command The command to be sent to the server.

These parameters may be obtained by using the FindByAlias() and FindByVO() methods.

RETURNS
The return value is `true` if everything went well, `false` otherwise. In the latter case, the error field becomes significant, and it may assume the following values.

VERR_NOSOCKET	The client was unable to connect to the server.
VERR_COMM	Some communication errors (Usually related to certificate problems)
VERR_SERVERCODE	The server returned an error code. More detailed information may be obtained by the ServeError() function.
VERR_PARAM	There was something wrong with the parameters passed to the function, or some of the required information (holder, etc...) is empty.
VERR_FORMAT	If the format of the data is unknown (e.g. neither an AC nor an old-style blob.
VERR_NOIDENT	If it was impossible to discover the holder of the AC or the client was unable to find its own proxy certificate.
VERR_NOINIT	The vomsdata object hasn't been properly initialized. Most likely the voms_dir and ca_dir parameters are empty.
VERR_PARSE	There has been some problem in parsing the AC or blob.
VERR_VERIFY	It was not possible to verify the signature.
VERR_SERVER	It was not possible to properly identify the Attribute Issuer.
VERR_TIME	The check on the validity dates failed.
VERR_IDCHECK	The holder of the AC is not the same entity as the holder of the enclosing certificate.

SEE ALSO
FindByAlias, FindByVO

14.3.20 BOOL CONTACTRAW(STD::STRING HOSTNAME, INT PORT, STD::STRING SERVSUBJECT, STD::STRING COMMAND, STD::STRING &RAW, INT &VERSION, INT TIMEOUT=-1)

This function is used to contact a specified server and use the received AC to fill the vomsdata structure.
hostname The fully qualified hostname of the machine on which the server runs.
port The port number on which the server is listening.
servsubject The subject of the server' certificate.
command The command to be sent to the server.
raw This is an output parameter, and it will contain the data received by the server.
version This, too, is an output parameter, and it will contain the version number of the data included.
timeout The timeout on the socket connection, in seconds. -1 means unlimited. The first four parameters may be obtained by using the FindByAlias() and FindByVO() methods.

RETURNS
The return value is `true` if everything went well, `false` otherwise. In the latter case, the error field becomes significant, and it may assume the following values.

VERR_NOSOCKET	The client was unable to connect to the server.
VERR_COMM	Some communication error (Usually related to certificate problems)
VERR_SERVERCODE	The server returned an error code. More detailed information may be obtained by the ServeError() function.
VERR_PARAM	There was something wrong with the parameters passed to the function, or some of the required information (holder, etc...) is empty.
VERR_FORMAT	If the format of the data is unknown (e.g. neither an AC nor an old-style blob.
VERR_NOIDENT	If the client was unable to find its own proxy certificate.
VERR_NOINIT	The vomsdata object hasn't been properly initialized. Most likely the voms_dir and ca_dir parameters are empty.

SEE ALSO
FindByAlias, FindByVO

14.3.21 BOOL CONTACTRESTRAW(CONST STD::STRING& HOSTNAME, INT PORT, CONST STD::STRING& COMMAND, STD::STRING& RAW, INT VERSION, INT TIMEOUT);

This function uses the REST protocol to contact voms. In everything else it is completely equivalent to ContactRaw(). Look at that function for explanation of the parameters and of the result.

SEE ALSO
ContactRaw

14.3.22 BOOL EXPORT(STD::STRING &DATA)

This function is used to create a string representation of all the data that has been read from VOMS certificates so far.
data This is an output parameter, and it will contain the data in encoded format.

RETURNS
The return value is `true` if everything went well, `false` otherwise. In the latter case, the error field becomes significant, and it may assume the following values.

VERR_MEM	There is not enough memory free.
VERR_FORMAT	There is an inconsistency in the internal data.
VERR_TYPE	The same as above. The difference is only for debugging purposes.

SEE ALSO
Import()

14.3.23 BOOL IMPORT(STD::STRING BUFFER)

This function is used to add a string created by the Export() call back into the vomsdata structure. This function also runs verification again.
buffer The string to convert.

RETURNS
The return value is `true` if everything went well, `false` otherwise. In the latter case, the error field becomes significant, and it may assume the following values:

VERR_PARAM	There was something wrong with the parameters passes to the function, or some of the required information (holder, etc...) is empty.
VERR_FORMAT	If the format of the data is unknown (e.g. neither an AC nor an old-style blob.
VERR_NOIDENT	If is was impossible to discover the holder of the AC or there was not a user certificate ready.
VERR_NOINIT	The vomsdata object hasn't been properly initialized. Most likely the voms_dir and ca_dir parameters are empty.
VERR_PARSE	There has been some problem in parsing the AC or blob.
VERR_VERIFY	It was not possible to verify the signature.
VERR_SERVER	It was not possible to properly identify the Attribute Issuer.
VERR_TIME	The check on the validity dates failed.
VERR_IDCHECK	The holder of the AC is not the same entity as the holder of the enclosing certificate.

14.3.24 BOOL VOMSDATA::DEFAULTDATA(VOMS &D)

This function returns the default attributes from a vomsdata class.
d This is the `voms` structure that will contain the default attributes.

RETURNS
The return value is `true` if everything went well, `false` otherwise. In the latter case, the error field becomes significant, and it may assume the following values:

VERR_NOEXT	If there was no default attributes (most likely because no attributes were read in.

14.3.25 STD::STRING VOMSDATA::ERRORMESSAGE(VOID)

```
std::string vomsdata::ErrorMessage(void);
```

This function returns the error message associated with the error code returned by the latest executed method.

RETURNS
Ther error message itself. Please note that this is completely meaningless if the latest method did not return an error code.

15 C API

There are three basic data structures: `data`, `voms` and `vomsdata`.

15.1 THE DATA STRUCTURE

The first one, `data` contains the data regarding a single attribute, giving its specification in terms of Groups, Roles and Capabilities. It is defined as follows:

```
struct data {
  char *group;
  char *role;
  char *cap;
};
```

All the values of these strings must be composed from regular expression: `a-ZA-Z0-9_/]*`.

15.1.1 GROUP

This field contains the name of a group into which the user belongs. The format of entries in this group is reminiscent of the structure of pathnames, and is the following:

/group/group/.../group

where the name of the first group is by convention the name of the Virtual Organization (VO), while each other */group* component is a subgroup of the group immediately preceding it on the left. The character '/' is not acceptable as part of a group name.

This field MUST always be filled.

15.1.2 ROLE

This field contains the name of the role to which the user owns in the group specified by `group`. If the user does not own any particular role in that group, than this field contains the value "NULL".

Compatibility

Or it may be set to `NULL`.

15.1.3 CAP

This field details a capability that the user has as a member of the group specified by `group` while owning the role specified by `role`. If there is no specific capability, than this value is "NULL".

Compatibility

Or it may be set to `NULL`.

No specific format is associated to a capability. They are basically free-form strings, whose value should be agreed between the AA and the Attribute verifier.

They are deprecated. No application should ever rely on them.

15.2 THE VOMS STRUCTURE

The second one, `voms` is used to group together all the informations that can be gleaned from a single AC, and is defined as follows:

```
#define  TYPE_NODATA 0  /*!< no data */
#define  TYPE_STD    1  /*!< group, role, capability triplet */
#define  TYPE_CUSTOM 2  /*!< result of an S command */


struct voms {
  int siglen;
  char *signature;
  char *user;
  char *userca;
  char *server;
  char *serverca;
  char *voname;
  char *uri;
  char *date1;
  char *date2;
  int   type;
  struct data **std;
  char *custom;
  int datalen;
  int version;
  char **fqan;
  char *serial;
  /* Fields below this line are reserved. */
};
```

The purpose of this structure is to present, in a readable format, the data that has been included in a single Attribute Certificate (AC). While the various public fields may be freely modified to simplify internal coding, such changes have no effect on the underlying AC. Let's examine the various fields in detail, starting with the constructors.

15.2.1 VERSION

This field specifies the version of this structure that is currently being used. A value of 0 indicates that it comes from an old format extension, while a value of 1 indicates that this structure comes from an AC.

Compatibility

As of VOMS 1.6.0, version 0 is no longer supported. What this means is that VOMS 1.6.0 or later is no longer compatible with VOMS 1.1.x or earlier.

15.2.2 SIGLEN

The length of the data signature.

15.2.3 USER

This field contains the subject of the holder's certificate in slash-separated format.

15.2.4 USERCA

This field contains the subject of the CA that issued the holder's certificate, in slash-separated format.

15.2.5 SERVER

This field contains the subject of the certificate that the AA used to issue the AC, in slash-separated format.

15.2.6 SERVERCA

This field contains, in slash-separated format, the subject of the CA that issued the certificate that the AA used to issue the AC.

15.2.7 VONAME

This field contains the name of the Virtual Organization (VO) to which the rest of the data contained in this structure applies.

15.2.8 URI

This is the URI at which the AA that issued this particular AC can be contacted. Its format is:

fqdn:*port*

where *fqdn* is the Fully Qualified Domain Name of the server which hosts the AA, while *port* is the port at which the AA can be contacted on that server.

15.2.9 DATE1, DATE2

These are the dates of start and end of validity of the rest of the informations. They are in a string representation readable to humans, but they may be easily converted back to their original format.

Here follows a code example doing that conversion:

```
ASN1_TIME *
convtime(char *data)
{
  char *data2 = strdup(data);

  if (data2) {
    ASN1_TIME *t= ASN1_TIME_new();
```

```
    t->data   = (unsigned char *)data2;
    t->length = strlen(data);
    switch(t->length) {
      case 15:
      t->type = V_ASN1_GENERALIZEDTIME;
      break;
      default:
      ASN1_TIME_free(t);
      return NULL;
    }
    return t;
  }
  return NULL;
}
```

15.2.10 TYPE

This datum specifies the type of data that follows. It can assume the following values:

TYPE_NODATA OBSOLETE

TYPE_CUSTOM OBSOLETE

Compatibility

Both TYPE_NODATA and TYPE_CUSTOM were only present in version 0. Since that version is no longer supported, their content is undocumented, and MUST not be used by applications.

TYPE_STD The data will contain (group, role, capabilities) triples.

15.2.11 STD

This vector contains all the attributes found in an AC, in the exact same order in which they were found, in the format specified by the `data` structure. It is only filled if the value of the `type` field is `TYPE_STD`.

15.2.12 CUSTOM

This field contains the data returned bu the "S" server command, and it is only filled if the `type` value id `TYPE_CUSTOM`.

15.2.13 FQAN

This field contains the same data as the `std` field, but specified in the Fully Qualified Attribute Name (FQAN) format.

15.2.14 VOMSDATA

The purpose of this object is to collect in a single place all informations present in a VOMS extension. All the fields should be considered read-only. Changing them has indefinite results.

```
struct vomsdata {
  char *cdir;
  char *vdir;
  struct voms **data;
  char *workvo;
  char *extra_data;
  int volen;
  int extralen;
  /* Fields below this line are reserved. */
};
```

Let us see the fields in detail.

15.2.15 DATA

This field contains a vector of `voms` structures, in the exact same order as the corresponding ACs appeared in the proxy certificate, and containing the informations present in that AC.

15.2.16 WORKVO, VOLEN

Compatibility

This fields is obsolete in the current version. Expect `workvo` to be set to `NULL` and `volen` to be set to 0.

15.2.17 EXTRA_DATA, EXTRALEN

This field contains additional data that has been added by the user via to the proxy via the `--include` command line option. `extralen` represents the length of that data.

15.2.18 CDIR, VDIR

This fields contain the paths, respectively, of the CA certificates and of the VOMS servers certificates.

15.3 FUNCTIONS

15.3.1 GENERALITIES

Most of these functions share two parameters, `struct vomsdata *vd`, and `int *error`. To avoid repetition, these two parameters are described here.
error This field contains the error code returned by one of the methods. Please note that the value of this field is only significant if the *last* method called returns an error value. Also, the value of this field

is subject to change without notice during method executions, regardless of whether an error effectively occurred.

The possible values returned are: VERR_NONE, VERR_NOSOCKET, VERR_NOIDENT, VERR_COMM, VERR_PARAM, VERR_NOEXT, VERR_NOINIT, VERR_TIME, VERR_IDCHECK, VERR_EXTRAINFO, VERR_FORMAT, VERR_NODATA, VERR_PARSE, VERR_DIR, VERR_SIGN, VERR_SERVER, VERR_MEM, VERR_VERIFY, VERR_TYPE, VERR_ORDER, VERR_SERVERCODE, VERR_NOTAVAIL

In general, a first idea of what each code means can be gleaned from the code name, but in any case every method description will document which errors its execution may generate and on which conditions.
vd This parameter is a pointer to the vomsdata structure that should be used by the function for both configuration and data retrieval and also for data storage.

15.3.2 STRUCT CONTACTDATA **VOMS_FINDBY*(STRUCT VOMSDATA *VD, CHAR *ALIAS, CHAR *SYSTEM, CHAR *USER, INT *ERROR)

```
struct contactdata **VOMS_FindByAlias(struct vomsdata *vd, char *
   alias, char *system, char *user, int *error);
struct contactdata **VOMS_FindByVO(struct vomsdata *vd, char *vo,
   char *system, char *user, int *error);
```

These two functions look in the vomses files installed in both the system-wide and user-specific directories for servers that have been registered with a particular alias or VO respectively.

The `contactdata` structure is defined below:

```
struct contactdata { /*!< You must never allocate directly this
   structure. Its
                        sizeof() is subject to change without
                           notice. The
                        only supported way to obtain it is via the
                                    VOMS_FindBy* functions. */
  char *nick;     /*!< The alias of the server */
  char *host;     /*!< The hostname of the server */
  char *contact;  /*!< The subject of the server's certificate */
  char *vo;       /*!< The VO served by this server */
  int   port;     /*!< The port on which the server is listening */
  char *reserved; /*!< HANDS OFF! */
};
```

alias The alias that will be searched for. The search will be case sensitive.
vo The vo that will be searched for. The search will be case sensitive.
system The directory where the system-wide files are located. If NULL then its default is /etc/vomses.
user The directory where the user-specific files are stored. If this field is NULL, then the default of $VOMS_USERCONF is used. If this is also empty, then the default become $HOME. **RETURNS**
The return value is a NULL-terminated vector containing the data (in `contactdata` format) of all the servers known by the system that go by the specified alias. This may be NULL if there was an error or no server was found registered with the specified alias.

The errors that you may find are:

VERR_MEM	Not enough memory.
VERR_DIR	There were some problems while traversing the directory.
VERR_NONE	No error occurred. Simply, no servers were found.

15.3.3 VOID VOMS_DELETECONTACTS(STRUCT CONTACTDATA **LIST)

This function deletes a vector of server data returned by either the `VOMS_FindByAlias{}` or the `VOMS_FindByVO()` functions. This is the only supported way to deallocate the vector. Any other attempt will result in undefined behavior.

It is although possible to deallocate only part of a vector. See the following code for an example.

```
/*
 * Supposing that v is a vector returned by one of the VOMS_FindBy*()
 * functions.  Also suppose that n is the vector's size (including
    the
 * NULL ending element).
 *
 * The following snippet will delete just the first member.
 */
struct contactdata *dummy[2];

dummy[1] = NULL;
dummy[0] = v[0];
v[0]     = v[n-1];
v[n-1]   = NULL;
VOMS_DeleteContacts(dummy);
```

list The data to be deleted.

RETURNS
None.

15.3.4 STRUCT VOMSDATA *VOMS_INIT(CHAR *VOMS, CHAR *CERT)

This function allocates and initializes a `vomsdata` structure. This is the only way to do so. Trying to allocate a `vomsdata` structure by any other way will trigger undefined behavior, since the structure that is published is only a small part of the real one.
voms The directory that contains the certificates of the VOMS servers. If this value is NULL, then $X509_VOMS_DIR is considered. If this is also empty than its default is .
cert The directory that contains the certificates of the CAs recognized by the server. If this value is NULL, then $X509_CERT_DIR is considered. If this is also empty than its default is: "/oetc/grid-security/certificates".

RETURNS
A pointer to a properly initialized `vomsdata` structure, or NULL if something went wrong. This is the only case in which an error code would no be associated to the function.

The default values are strongly suggested. If you want to hardcode specific ones, think very hard about the less of configurability that it would entail.

15.3.5 STRUCT VOMS *VOMS_COPY(STRUCT VOMS *, INT *ERROR)

This function duplicates an existing `voms` structure. It is the *only* supported way to do so.
voms The voms structure that you wish to be duplicated.

RETURNS
A pointer to a voms structure that duplicates the content of the one you passed, or NULL if something went wrong.

ERRORS
VERR_MEM Not enough memory.

15.3.6 STRUCT VOMSDATA *VOMS_COPYALL(STRUCT VOMSDATA *VD, INT *ERROR)

This function duplicates an existing `vomsdata` structure. It is the *only* supported way to do so.

RESULTS
A pointer to a voms structure that duplicates the content of the one you passed, or NULL if something went wrong.

ERRORS
VERR_MEM Not enough memory.

15.3.7 VOID VOMS_DELETE(STRUCT VOMS *V)

This functions deletes an existing `voms` structure. It is the ONLY supported way to do so.
v A pointer to the voms structure to delete. It is safe to call this function with a NULL pointer.

RESULTS
None.

15.3.8 INT VOMS_ADDTARGET(STRUCT VOMSDAA *VD, CHAR *TARGET, INT *ERROR)

This function adds a target to the target list for the AC that will be generated by a server when it will be contacted by the `VOMS_Contact*()` function.
target The target to add. It should be a Fully Qualified Domain Name.

RESULTS

0 If something went wrong.

<>**0** Otherwise.

ERRORS

VERR_NOINIT	The `vomsdata` structure was not properly initialized.
VERR_PARAM	The `target` parameter was NULL.
VERR_MEM	There was not enough memory.

15.3.9 VOID VOMS_FREETARGETS(STRUCT VOMSDATA *VD , INT *ERROR)

This function resets the list of targets. It always succeeds. It is also safe to call this function when targets have been set.

15.3.10 CHAR *VOMS_LISTTARGETS(STRUCT VOMSDATA *VD, INT *ERROR)

This function returns a comma separated string containing all the targets that have been set by the `VOMS_AddTarget()` function. The caller is the owner of the returned string, and is responsible for calling `free()` over it when he no longer needs it.

RESULTS
A string with the result, or NULL.

VERR_NOINIT	The `vomsdata` structure was not properly initialized.
VERR_MEM	There was not enough memory.

15.3.11 INT VOMS_SETVERIFICATIONTYPE(INT TYPE, STRUCT VOMSDATA *VD, INT *ERROR)

This function sets the type of AC verification done by the `VOMS_Retrieve()` and `Contact()` functions. The choices are detailed in the `verify\_type` type.

```
#define VERIFY_FULL       0xffffffff
#define VERIFY_NONE       0x00000000
#define VERIFY_DATE       0x00000001
#define VERIFY_NOTARGET   0x00000002
#define VERIFY_KEY        0x00000004
#define VERIFY_SIGN       0x00000008
#define VERIFY_ORDER      0x00000010
#define VERIFY_ID         0x00000020
#define VERIFY_CERTLIST   0x00000040
```

The meaning of these types is the following:

VERIFY_DATE This flag verifies that the current date is within the limits specified by the AC itself.

VERIFY_TARGET This flag verifies that the AC is being evaluated in a machine that is included in the target extension of the AC itself.

VERIFY_KEY This flag is for a future extension and is unused at the moment.

VERIFY_SIGN This flag verifies that the signature of the AC is correct.

VERIFY_ORDER This flag verifies that the attributes present in the AC are in the exact order that was requested. Please note that this can ONLY be done when examining an AC right after generation with the Contact() function. This flag is meaningless in all other cases.

VERIFY_ID This flag verifies that the holder information present in the AC is consistent with:

1. The enveloping user proxy in case the AC was contained in one.

2. The user's own certificate in case the AC was received without an enclosing proxy.

VERIFY_CERTLIST This flag verifies the issuer's certificate included in the AC, if present.

VERIFY_FULL This flag implies all other verifications.

VERIFY_NONE This flag disables all verifications.

These flags can be combined by OR-ing them together. However, if VERIFY_NONE is OR-ed to any other flag, it can be dismissed, while if VERIFY_FULL is OR-ed to any other flag, all other flags ca be dismissed.

If this function is not explicitly called by the user, a VERIFY_FULL flag is in effect.

RESULTS

0 If there is an error.

<> **0** otherwise.

VERR_NOINIT	The `vomsdata` structure was not properly initialized.

15.3.12 INT VOMS_SETLIFETIME(INT LENGTH, STRUCT VOMSDATA *VD, INT *ERROR)

```
int VOMS_SetLifetime(int length, struct vomsdata *vd, int *error);
```

This function sets the requested lifetime for ACs that would be generated as the result of a `VOMS_Contact()` or `VOMS_ContactRaw()}` request. Note however that this is only an hint sent to the server, since it can lower it at will if the requested length is against server policy.

length The lifetime requested, measured in seconds.

RESULTS

0 If there is an error.

<> **0** otherwise.

VERR_NOINIT	The `vomsdata` structure was not properly initialized.

SEE ALSO

VOMS_Contact(), VOMS_ContactRaw()

15.3.13 VOID VOMS_DESTROY(STRUCT VOMSDATA *VD)

```
void VOMS_Destroy(struct vomsdata *vd);
```

This function destroys an allocated `vomsdata` structure. It is the *only* supported way to do so. It is also safe to pass a NULL pointer to it.

RESULTS
None.

15.3.14 INT VOMS_ORDERING(CHAR *ORDER, STRUCT VOMSDATA *VD, INT *ERROR)

```
int VOMS_Ordering(char *order, struct vomsdata *vd, int *error);
```

This function is used to request a specific ordering of the attributes present in an AC returned by the `VOMS_Contact()` or by the `VOMS_ContactRaw()` functions.

This function can be called several times, each time specifying a new attribute. The attributes in th AC created by the server will be in the same order as the calls to this function, ignoring attributes specified by this function that the server does not wish to grant. Attributes not explicitly specified in this list will be inserted, in an unspecified order, after all the others.

Never calling this function means that the corresponding list will be empty, and as a consequence all the attributes will be in an unspecified ordering.
order The name of an attribute, in either the FQAN format (i.e: /group/Role=role) or in the <group>[:<role>: format.

Compatibility

Not all versions of the APIs support the FQAN format yet.

RESULTS

0 If there is an error.

<> **0** otherwise.

ERRORS

VERR_NOINIT	The `vomsdata` structure was not properly initialized.
VERR_PARAM	The `order` parameter is NULL.
VERR_MEM	There is not enough memory.

SEE ALSO
VOMS_ResetOrder(), VOMS_Contact(), VOMS_ContactRaw()

15.3.15 INT VOMS_RESETORDER(STRUCT VOMSDATA *CD, INT *ERROR)

```
int VOMS\_ResetOrder(struct vomsdata *cd, int *error);
```

This function resets the attribute ordering set by the `VOMS_Ordering` function.

RESULTS

0 If there is an error.

<> **0** otherwise.

VERR_NOINIT The `vomsdata` structure was not properly initialized.

SEE ALSO
VOMS_Ordering()

15.3.16 INT VOMS_CONTACT(CHAR *HOSTNAME, INT PORT, CHAR *SERVSUBJECT, CHAR *COMMAND, STRUCT VOMSDATA *VD, INT *ERROR)

```
int VOMS_Contact(char *hostname, int port, char *servsubject, char *
    command, struct vomsdata *vd, int *error);
```

This function is used to contact a VOMS server to receive an AC containing the calling user's authorization informations. A prerequisite to calling this function is the existence of a valid proxy for the user himself. This function does not create such a proxy, which then must already exist. Also, the parameters needed to call this function should have been obtained by calling one of `FindByAlias()` or `FindByVO()`.
hostname This is the hostname of the machine hosting the server.
port This is the port number on which the server is listening.
servsubject This is the subject of the VOMS server' certificate. This is needed for the mutual authentication.
command This is the command to be sent to the server. For more info about it, consult the voms-proxy-init() manual.

RESULTS

0 If there is an error.

<>**0** otherwise. Furthermore, the data returned by the server has been parsed and added to the `vomsdata` structure.

ERRORS

VERR_NOINIT	If the vomsdata structure was not properly initialized.
VERR_NOSOCKET	If it was impossible to contact the server.
VERR_MEM	If there was not enough memory.
VERR_IDCHECK	If a proxy certificate was not found or the data returned by the server did not contain identifying information.
VERR_FORMAT	If there was an error in the format of the data received.
VERR_NODATA	If no data was received at all. (Usually as a consequence of either a server error or not being recognized by the server as a valid user.)
VERR_ORDER	If the attribute that the client requested, via the `VOMS_Ordering()` function, to be first in the list of attributes received is not first in the attributes returned by the server. This particular code means that the data has been correctly interpreted and is available in the vomsdata structure if you want to use it.
VERR_SERVERCODE	Some strange error occurred in the server.

SEE ALSO
VOMS_FindByVO(), VOMS_FindByAlias()

15.3.17 INT VOMS_CONTACTRAW(CHAR *HOSTNAME, INT PORT, CHAR *SERVSUBJECT, CHAR *COMMAND, VOID **DATA, INT *DATALEN, INT *VERSION, STRUCT VOMSDATA *VD, INT *ERROR)

This function, like `VOMS_Contact()` can be used to contact a server and receive Authorization info from it. The difference between the two functions is that this version does not interpret the raw data, but on the contrary returns it to the caller. This function has all the same prerequisites as `VOMS_Contact()`.
hostname This is the hostname of the machine hosting the server.
port This is the port number on which the server is listening.
servsubject This is the subject of the VOMS server' certificate. This is needed for the mutual authentication.
command This is the command to be sent to the server. For more info about it, consult the voms-proxy-init() manual.
data A pointer to a pointer to an area of memory where the data returned from the server is stored. It is the caller's responsibility to free() this memory when it is no longer useful.
datalen The length of the data returned.
version The version of the AC returned. Note that this is a *minimum* version, it only guarantees that the data is *at least* in that version of the format.

RESULTS

0 If there is an error.

<>**0** otherwise. Furthermore, the data returned by the server has been parsed and added to the `vomsdata` structure.

ERRORS

VERR_NOINIT	If the vomsdata structure was not properly initialized.
VERR_NOSOCKET	If it was impossible to contact the server.
VERR_MEM	If there was not enough memory.
VERR_IDCHECK	If a proxy certificate was not found or the data returned by the server did not contain identifying information.
VERR_FORMAT	If there was an error in the format of the data received.
VERR_NODATA	If no data was received at all. (Usually as a consequence of either a server error or not being recognized by the server as a valid user.)
VERR_ORDER	If the attribute that the client requested, via the `VOMS_Ordering()` function, to be first in the list of attributes received is not first in the attributes returned by the server. This particular code means that the data has been correctly interpreted and is available in the vomsdata structure if you want to use it.
VERR_SERVERCODE	Some strange error occur ed in the server.

15.3.18 INT VOMS_RETRIEVE*(X509 *CERT, STACK_OF(X509) *CHAIN, INT HOW, STRUCT VOMSDATA *VD, INT *ERROR)

```
int VOMS_Retrieve(X509 *cert, STACK_OF(X509) *chain, int how, struct
   vomsdata *vd, int *error);
int VOMS_RetrieveEXT(X509_Extension *ext, struct vomsdata *vd, int *
   error);
int VOMS_RetrieveFromCred(gss_cred_id_t cred, int how, struct
   vomsdata *vd, int *error);
int VOMS_RetrieveFromCtx(gss_ctx_id_t cred, int how, struct vomsdata
    *vd, int *error);
int VOMS_RetrieveFromProxy(int how, struct vomsdata *vd, int *error);
int VOMS_RetrieveFromFile(FILE *file, int how, struct vomsdata *vd,
   int *error);
```

This functions are used to extract from a credential the VOMS-specific extension, to parse them and to insert the results into the `vomsdata` structure. Each variant of it extracts information from their specific type of credential.

cert This is the certificate that contains the VOMS information. No checks are done on the validity of this certificate, that is supposed to have already been verified by some other means.

chain This is the chain of certificates that signed the cert certificate. This pointer may be null, but see the next parameter.

context A GSS context. This is created in a server when a GSS authenticated connection occurs and includes the other side's credentials.

credential A GSS credential.

ext An X509 extension, which should be the extension used by VOMS to include the AC.

file A file from which the proxy will be loaded.
how This parameter indicates how the search for the VOMS info will be performed. If RECURSE_CHAIN then the information is searched first into the cert and then, if it was not found, in the walking the chain, from the certificates to the CA. If RECURSE_NONE is specified, then the information is only searched in the cert. In case the first value is specified, then the searches stop as soon as the info is found, ignoring further extension that may be found down the chain.

RESULTS

0 If there is an error.

<>**0** otherwise. Furthermore, the data returned by the server has been parsed and added to the `vomsdata` structure.

ERRORS

VERR_NOINIT	If the vomsdata structure was not properly initialized.
VERR_PARAM	If there is something wrong with one of the parameters.
VERR_MEM	If there was not enough memory.
VERR_IDCHECK	If a proxy certificate was not found or the data returned by the server did not contain identifying information.
VERR_FORMAT	If there was an error in the format of the data received.
VERR_NOEXT	If the extension was not found.
VERR_NOTAVAIL	If the `RetrieveFromCred()`, `RetrieveFromCtx()` or `RetrieveFromProxy()` functions were used with the version of the libraries which add globus support disabled.

15.3.19 INT VOMS_IMPORT(CHAR *BUFFER, INT BUFLEN, STRUCT VOMSDATA *VD, INT *ERROR)

```
int VOMS_Import(char *buffer, int buflen, struct vomsdata *vd, int *
    error);
```

This function is used to add a string created with `VOMS_Export()` back into the vomsdata structure.
buffer A pointer to the string.
buflen The length of the string.

RESULTS

0 If there is an error.

<>**0** otherwise. Furthermore, the data returned by the server has been parsed and added to the `vomsdata` structure.

ERRORS

VERR_NOINIT	If the vomsdata structure was not properly initialized.
VERR_FORMAT	If there was an error in the format of the data received.
VERR_PARAM	If there is something wrong with one of the parameters.
VERR_MEM	If there was not enough memory.
VERR_IDCHECK	If a proxy certificate was not found or the data returned by the server did not contain identifying information.
VERR_SERVER	The VOMS server was unidentifiable.
VERR_PARSE	There has been some problem in parsing the AC or blob.
VERR_SIGN	It was not possible to verify the signature.
VERR_SERVER	It was not possible to properly identify the Attribute Issuer.
VERR_TIME	The check on the validity dates failed.

15.3.20 INT VOMS_EXPORT(CHAR **BUFFER, INT *BUFLEN, STRUCT VOMSDATA *VD, INT *ERROR)

```
int VOMS_Export(char **buffer, int *buflen, struct vomsdata *vd, int
    *error);
```

This function will take the current `vomsdata` structure and encode it in a string that can then be exported.
buffer A pointer to an area of memory that will be allocated and filled by the function. It is the caller's responsibility to free() this memory. It is possible that this pointer will be set to NULL, in case the vomsdata structure is empty.
buflen The size of the data pointed by buffer.

RESULTS

0 If there is an error.

<>**0** otherwise. Furthermore, the data returned by the server has been parsed and added to the `vomsdata` structure.

ERRORS

VERR_PARAM	If there is something wrong with one of the parameters.
VERR_MEM	If there was not enough memory.

15.3.21 STRUCT VOMS *VOMS_DEFAULTDATA(STRUCT VOMSDATA *VD, INT *ERROR)

```
struct voms *VOMS_DefaultData(struct vomsdata *vd, int *error);
```

This function returns the default attributes from a vomsdata class.

RESULTS

NULL There has been an error or the `vomsdata` structure was empty.

<>**NULL** There is some data.

ERRORS

VERR_NOINIT	The `vomsdata` structure was not properly initialized.
VERR_NONE	The `vomsdata` structure was empty.

15.3.22 CHAR *VOMS_ERRORMESSAGE(STRUCT VOMSDATA *VD, INT *ERROR, CHAR *BUFFER, INT LEN)

```
char *VOMS_ErrorMessage(struct vomsdata *vd, int *error, char *buffer
    , int len);
```

This function returns an error message describing the error of the last function called. It is significant if and only if the last call returned an error, otherwise the returned string will be meaningless.
buffer A pointer to a location of memory where the memory message will be written. This may be NULL, in which case the message will be written in memory newly allocated by this function.
len The length of the previous buffer. This is only significant if `buffer` is not NULL

RESULTS
A pointer to a string describing the error. If the passed `buffer` was NULL, then the caller is responsible for `free()`ing this memory when it is no longer needed.

15.3.23 INT VOMS_GETATTRIBUTESOURCEHANDLE(STRUCT VOMS *V, INT NUM, STRUCT VOMSDATA *VD, INT *ERROR)

```
int VOMS_GetAttributeSourceHandle(struct voms *v, int num, struct
    vomsdata *vd, int *error);
```

This function returns an handle to a specific attribute container.

v The voms structure from which attributes are to be extracted.
num The number of the desired container. See VOMS_GetAttributeSourcesNumber() to get the maximum value.

RESULTS
An handle to the container, or -1 if the specified container does not exist.

ERRORS

VERR_PARAM	If the specified container does not exist..

SEE ALSO
VOMS_GetAttributeSourcesNumber()

15.3.24 VOMS_GETATTRIBUTESOURCESNUMBER(STRUCT VOMS *V, STRUCT VOMSDATA *VD, INT *ERROR)

```
VOMS_GetAttributeSourcesNumber(struct voms *v, struct vomsdata *vd,
   int *error);
```

The Generic Attributes that may be present in an AC are organized by grantor. This function returns the number of grantors.

v The voms structure from which attributes are to be extracted.

RESULTS
The number of grantors, 0 if Generic Attributes were not present, or -1 in case of errors.

ERRORS
VERR_PARAM If an error is present in the format of the AC.

15.3.25 VOMS_GETATTRIBUTEGRANTOR(STRUCT VOMS *V, INT HANDLE, STRUCT VOMSDATA*VD, INT *ERROR)

```
VOMS_GetAttributeGrantor(struct voms *v, int handle, struct vomsdata*
   vd, int *error);
```

This function will return the name of the attributes grantor of the attribute block identified by the handle. This memory belongs to the function, and MUST NOT be tampered with.

v The voms structure from which attributes are to be extracted.
handle The handle to the attribute container as returned by VOMS_GetAttributeSourceHandle()

RESULTS
The string representing the grantor, or NULL in case of errros.

ERRORS
VERR_PARAM If the specified container does not exist, or another error is present in the format of the AC.

15.3.26 VOMS_GETATTRIBUTESNUMBER(STRUCT VOMS *V, INT HANDLE, STRUCT VOMSDATA *VD, INT *ERROR)

```
VOMS_GetAttributesNumber(struct voms *v, int handle, struct vomsdata
    *vd, int *error);
```

This function will return the number of attributes present in a container.

v The voms structure from which attributes are to be extracted.
handle The handle to the attribute container as returned by VOMS_GetAttributeSourceHandle()

RESULTS
The number of attributes, or -1 in case of errors.

ERRORS
VERR_PARAM If an error is present in the format of the AC.

15.3.27 VOMS_GETATTRIBUTE(STRUCT VOMS *V, INT HANDLE, INT NUM, STRUCT ATTRIBUTE *A, STRUCT VOMDATA *VD, INT *ERROR)

```
VOMS_GetAttribute(struct voms *v, int handle, int num, struct
   attribute *a, struct vomdata *vd, int *error);
```

This function extracts the selected attribute, and fills the passed `attribute` structure with the relevan information.

Format of the `attribute` structure:

```
struct attribute {
  char *name;
  char *value;
  char *qualifier;
};
```

Where `name` is the name of the attribute, `value` is its value, and `qualifier` is an optional qualifier, which may be NULL if not passed.

v The voms structure from which attributes are to be extracted.
handle The handle to the attribute container as returned by VOMS_GetAttributeSourceHandle()
num The number of the attribute requested
a OUTPUT PARAMETER: The members of this structure get filled with the corresponding values from the attribute

RESULTS
1 in case of success, 0 otherwise.

ERRORS
VERR_PARAM If an error is present in the format of the AC.

15.3.28 VOMS_GETAC(STRUCT VOMS *V)

```
AC *VOMS_GetAC(struct voms *v);
```

This function is used to access the AC corresponding to the `v` structure passed. This is only a copy. Modifying this has no effect on the structure. The application is responsible for calling AC_Free() on it.

v The voms structure for which the AC is desired.

RESULTS
A pointer to the AC if successfull, NULL otherwise.

15.3.29 STRUCT VOMSDATA *VOMS_DUPLICATE(STRUCT VOMSDATA *VD)

This function duplicates a whole `vomsdata` structure.

vd The structure to duplicate

RESULTS
A pointer to the AC if successful, NULL otherwise.

15.3.30 INT VOMS_SETVERIFICATIONTIME(TIME_T VERIFICATIONTIME, STRUCT VOMSDATA *VD, INT *ERROR)

```
int VOMS_SetVerificationTime(time_t verificationtime, struct vomsdata
    *vd, int *error)
```

This function modifies the behaviour of the AC verification. The verification is done as if the time, rather than being the current time, was the time passed at this function. This is useful to verify, for example, if an AC will be valid 12 hours from now. Note that obviously this cannot predict certificate revocation, so the CRL used are still the ones available at the time when the verification is done.

Passing '0' as an argument disables this capability, and restores the current time as the time of verification.

verificationtime The time that will be considered 'current' for verification purposes.

RESULTS

1 if successful, 0 otherwise

ERRORS

VERR_PARAM If an error is present in the parameters.

15.3.31 CHAR **VOMS_GETTARGETSLIST(STRUCT VOMS *V, STRUCT VOMSDATA *VD, INT *ERROR)

```
char **VOMS_GetTargetsList(struct voms *v, struct vomsdata *vd, int *
    error)
```

This function returns an array of strings, NULL-terminated, containing the targets of the specified AC. VOMS_FreeTargetsList() must be called to free the array.

v The AC in question

RESULTS

A NULL-terminated array of targets, or NULL in case of problems

ERRORS

VERR_PARAM If an error is present in the parameters.

15.3.32 VOID VOMS_FREETARGETSLIST(CHAR **TARGETS)

```
void VOMS_FreeTargetsList(char **targets)
```

This function frees the array returned by VOMS_GetTargetsList(). This function can handle a NULL argument.

targets The array

RESULTS

The array gets freed.

15.3.33 INT VOMS_RETRIEVEEXT(X509_EXTENSION, STRUCT VOMSDATA *VD, INT *ERROR)

```
extern int VOMS_RetrieveEXT(X509_EXTENSION *ext, struct vomsdata
*vd, int *error);
```

This function retrieves the credentials from the passed extension.

ext The extension containing the VOMS AC.

RESULTS

0 If there is an error.

<>**0** otherwise. Furthermore, the data returned by the server has been parsed and added to the `vomsdata` structure.

15.3.34 INT VOMS_RETRIEVEFROMFILE(FILE *FILE, INT HOW, STRUCT VOMSDATA *VD, INT *ERROR)

```
extern int VOMS_RetrieveFromFile(FILE *file, int how, struct vomsdata
*vd, int *error);
```

This function retrieves the credentials from the passed extension.

file The file containing the VOMS proxy
how The strategy to use in parsing. See VOMS_Retrieve for the details if the value

RESULTS

0 If there is an error.

<>**0** otherwise. Furthermore, the data returned by the server has been parsed and added to the `vomsdata` structure.

SEE ALSO
VOMS_Retrieve

15.3.35 EXTERN INT VOMS_SETTIMEOUT(INT T, STRUCT VOMSDATA *VD, INT *ERROR);

```
extern int VOMS_SetTimeout(int t, struct vomsdata *vd, int *error);
```

This function sets the timeout to use when contacting the voms server.

t The timeout, in seconds. -1 means no timeout.

RESULTS
Always 1 (success)

15.3.36 EXTERN INT VOMS_LOADCREDENTIALS(X509 *CERT, EVP_PKEY *PKEY, STACK_OF(X509)* CHAIN, STRUCT VOMSDATA *VD, INT *ERROR);

```
extern int VOMS_LoadCredentials(X509 *cert, EVP\_PKEY *pkey, STACK\
   _OF(X509)* chain, struct vomsdata *vd, int *error);
```

This function loads the credentials from the specified locations to use them for connecting to the voms server.

cert The certificate to load.
key The private key associate to the certificate.
chain The certificate chain associated to the certificate.

RESULTS

0 If there is an error.

<>**0** otherwise.

15.3.37 INT PROXY_VERIFY_CALLBACK_SERVER(X509_STORE_CTX *CTX, VOID *DATA)

```
int proxy_verify_callback_server(X509_STORE_CTX *ctx, void *data)
```

This function should not be called directly. Instead, it should be setup as a callback vai the `SSL_CTX_set_verify` function during the setup of the SSL_CTX object if support for proxies over SSL is desired in a server.

15.3.38 INT PROXY_VERIFY_CALLBACK_CLIENT(INT OK, X509_STORE_CTX (CTX)

```
int proxy_verify_callback_client(int ok, X509_STORE_CTX (ctx)
```

This function should not be called directly. Instead, it should be setup as a callback vai the `SSL_CTX_set_verify` function during the setup of the SSL_CTX object if support for proxies over SSL is desired in a client.

15.3.39 VOID SETUP_SSL_PROXY_HANDLER(SSL *SSL, CHAR *CADIR)

```
void setup_SSL_proxy_handler(SSL *ssl, char *cadir)
```

After having setup the proper callback, and immediately after having created the SSL object from the context, call this function to activate proxy support on an SSL connection.

ssl The freshly created SSL object.
cadir The directory that contains the CA certificates

15.3.40 VOID DESTROY_SSL_PROXY_HANDLER(SSL *S)

```
void destroy_SSL_proxy_handler(SSL *s)
```

Immediately before disposing of a SSL object, call this function on it to release the extra memory used by the proxy setup. This method must be called even if the object was never used in a connection.

s The SSL object from which he proxy handler has to be freed.

16 JAVA APIS

Package org.glite.voms
Package Contents *Page*

Classes

16.1 CLASS BASICVOMSTRUSTSTORE

Deprecated! Use PKIStore instead.

16.1.1 DECLARATION

public final class BasicVOMSTrustStore
extends java.lang.Object
implements org.glite.voms.ac.ACTrustStore

16.1.2 FIELD SUMMARY

DEFAULT_TRUST_STORE_LISTING

16.1.3 CONSTRUCTOR SUMMARY

BasicVOMSTrustStore() Creates a default VOMS trust store.
BasicVOMSTrustStore(String, long) Creates and manages an in-memory cache of VOMS issuers by periodically scanning a directory containing the trusted issuers.

16.1.4 METHOD SUMMARY

getAACandidate(X500Principal)
getDirList()
refresh() Refreshes the in-memory cache of trusted signer certificates.
stopRefresh()

16.1.5 FIELDS

- public static final java.lang.String **DEFAULT_TRUST_STORE_LISTING**

16.1.6 CONSTRUCTORS

- **BasicVOMSTrustStore**
  `public` **BasicVOMSTrustStore**`( )`
  - **Description**
    Creates a default VOMS trust store. Equivalent to
    `new BasicVOMSTrustStore(DEFAULT_TRUST_STORE_LISTING, 300000);`
- **BasicVOMSTrustStore**
  `public` **BasicVOMSTrustStore**`( java.lang.String` **trustedDirList**`, long` **refreshPeriod** `)`
  - **Description**
    Creates and manages an in-memory cache of VOMS issuers by periodically scanning a directory containing the trusted issuers. If `refreshPeriod` is 0, it never refreshes.
  - **Parameters**
    * `trustedDirList` – directory listing containing trusted VOMS certs
    * `refreshPeriod` – refresh period in milliseconds
  - **See also**
    * `DirectoryList`

16.1.7 METHODS

- **getAACandidate**
  `java.security.cert.X509Certificate[]` **getAACandidate**`( javax.security.auth.x500.X500Principa`
  issuer `)`
  - **Description copied from ac.ACTrustStore** (**in 16.14, page 125**)
    Returns an array of issuer candidates, by performing a name comparison of the AC's issuer and the subject names of the certificates in the trust store.
    NOTE: No actual verification or validation of signature takes place in this function.
  - **Parameters**
    * `issuer` – the principal to find an issuer for. If `null`, all known AAs will be returned.
  - **Returns** – an array of issuer candidates, or `null` in case of an error.

- **getDirList**
  `public java.lang.String` **getDirList**`( )`
- **refresh**
  `public void` **refresh**`( )`
  - **Description**
    Refreshes the in-memory cache of trusted signer certificates.
- **stopRefresh**
  `public void` **stopRefresh**`( )`

16.2 CLASS FQAN

Parses and assembles Fully Qualified Attribute Names (FQANs) used by VOMS. FQANs are defined as `<group>[/Role=[<role>][/Capability=<capability>]]`

16.2.1 DECLARATION

public class FQAN
extends java.lang.Object

16.2.2 CONSTRUCTOR SUMMARY

FQAN(String)
FQAN(String, String, String)

16.2.3 METHOD SUMMARY

equals(Object)
getCapability()
getFQAN()
getGroup()
getRole()
split()
toString()

16.2.4 CONSTRUCTORS

- **FQAN**
  `public` **FQAN**`( java.lang.String` **fqan** `)`
- **FQAN**
  `public` **FQAN**`( java.lang.String` **group**`, java.lang.String` **role**`, java.lang.String` **capability** `)`

16.2.5 METHODS

- **equals**
  `public boolean` **equals**`( java.lang.Object )`
- **getCapability**
  `public java.lang.String` **getCapability**`( )`
- **getFQAN**
  `public java.lang.String` **getFQAN**`( )`
- **getGroup**
  `public java.lang.String` **getGroup**`( )`
- **getRole**
  `public java.lang.String` **getRole**`( )`
- **split**
  `protected void` **split**`( )`
- **toString**
  `public java.lang.String` **toString**`( )`

16.3 CLASS LSCFILE

The job of this class is to represent a *.lsc file in the vomsdir directory.

16.3.1 DECLARATION

public class LSCFile
extends java.lang.Object

16.3.2 CONSTRUCTOR SUMMARY

LSCFile(File) Loads a *.lsc file from a File

16.3.3 METHOD SUMMARY

getDNLists() Returns the allowed subject/issuer DN sequences for this file.
getName() Returns the basename of the file from which this was loaded.

16.3.4 CONSTRUCTORS

- **LSCFile**
  `public` **LSCFile**`( java.io.File` **f** `) throws java.io.IOException`
  - **Description**
    Loads a *.lsc file from a File

– **Parameters**
  * `f` – the file to load from
– **Throws**
  * `java.io.IOException` – if there are problems loading the file.

16.3.5 METHODS

- **getDNLists**
  `public java.util.Vector` **getDNLists**`( )`
  – **Description**
    Returns the allowed subject/issuer DN sequences for this file.
  – **Returns** – a vector whose elements are vectors of strings describing the exact sequences.
- **getName**
  `public java.lang.String` **getName**`( )`
  – **Description**
    Returns the basename of the file from which this was loaded.
  – **Returns** – the filename, or null if nothing was loaded.

16.4 CLASSPKISTOREFACTORY

PKIStoreFactory is the class used to create objects of type PKIStore.

16.4.1 DECLARATION

public class PKIStoreFactory
extends java.lang.Object

16.4.2 METHOD SUMMARY

getStore(String, int, boolean, boolean) gets a PKIStore
getStore(String, int, boolean) gets a PKIStore
getStore(String, int) gets a PKIStore
getStore(int) gets a PKIStore

16.4.3 METHODS

- **getStore**
  `public static org.glite.voms.PKIStore` **getStore**`( java.lang.String` **dir**`, int type, boolean` **aggressive**`, boolean timer)`
  – **Description**
    Creates a PKIStore object.

  - **Parameters**
    - `dir` – The directory from which to read the credentials
    - `type` – either PKIStore.TYPE_VOMS or PKIStore.TYPE_CA
    - `aggressive` – wether reading should be aggressive
    - `timer` – should try rereading after some time

- **getStore**
  `public static org.glite.voms.PKIStore` **getStore**`( java.lang.String` **dir**`, int type, boolean` **aggressive**`)`
  - **Description**
    Creates a PKIStore object.
  - **Parameters**
    - `dir` – The directory from which to read the credentials
    - `type` – either PKIStore.TYPE_VOMS or PKIStore.TYPE_CA
    - `aggressive` – wether reading should be aggressive

- **getStore**
  `public static org.glite.voms.PKIStore` **getStore**`( java.lang.String` **dir**`, int type)`
  - **Description**
    Creates a PKIStore object.
  - **Parameters**
    - `dir` – The directory from which to read the credentials
    - `type` – either PKIStore.TYPE_VOMS or PKIStore.TYPE_CA

- **getStore**
  `public static org.glite.voms.PKIStore` **getStore**`( int type)`
  - **Description**
    Creates a PKIStore object.
  - **Parameters**
    - `dir` – The directory from which to read the credentials
    - `type` – either PKIStore.TYPE_VOMS or PKIStore.TYPE_CA
    - `aggressive` – wether reading should be aggressive
    - `timer` – should try rereading after some time

16.5 CLASS PKISTORE

PKIStore is the class serving to store all the components of a common PKI installation, i.e.: CA certificates, CRLs, Signing policy files... It is also capable of storing files specific to the handling of VOMS proxies, i.e. the content of the vomsdir diectory.

16.5.1 DECLARATION

public class PKIStore
extends java.lang.Object
implements org.glite.voms.ac.VOMSTrustStore

16.5.2 FIELD SUMMARY

TYPE_CADIR This PKIStore object will contain data from a CA directory.
TYPE_VOMSDIR This PKIStore object will contain data from a vomsdir directory.

16.5.3 CONSTRUCTOR SUMMARY

PKIStore()
PKIStore(String, int) This is equivalent to PKIStore(dir, type, true)
PKIStore(String, int, boolean)

16.5.4 METHOD SUMMARY

getAACandidate(X500Principal, String) Gets an array of candidate issuer certificates for an AC with the given issuer and belonging to the given VO.
getCAs()
getCRLs()
getLSC(String, String) Gets the LSC file corresponding to the given VO, for the given server.
getSignings()
load() Loads the files from the directory specified in the constructors
refresh() Refreshes the content of the PKIStore object.
rescheduleRefresh(int) Changes the interval between refreshes of the store.
setAggressive(boolean) Changes the aggressive mode of the store.
stopRefresh() Stop all refreshes.

16.5.5 FIELDS

- public static final int **TYPE_VOMSDIR**
  - This PKIStore object will contain data from a vomsdir directory.
- public static final int **TYPE_CADIR**
  - This PKIStore object will contain data from a CA directory.

16.5.6 CONSTRUCTORS

- **PKIStore**
  `public` **PKIStore**`( )`
- **PKIStore**
  `public` **PKIStore**`( java.lang.String` **dir**`, int` **type** `) throws java.io.IOException, java.security`
  `java.security.cert.CRLException`

- **Description**

  This is equivalent to PKIStore(dir, type, true)

- **See also**

  * `PKIStore(java.lang.String,int,boolean)` (in 16.5.6, page 96)

- **PKIStore**

  `public` **PKIStore**`( java.lang.String` **dir**`, int` **type**`, boolean` **aggressive** `) throws java.io.IOExcep`
  `java.security.cert.CertificateException, java.security.cert.CRLException`

  - **Parameters**

    * `dir` – – The directory from which to read the files. If null or the empty string, this will default to "/etc/grid-security/certificates" if type is TYPE_CADIR, or "etc/grid-security/vomsdir" if type is TYPE_VOMSDIR.
    * `type` – – either TYPE_CADIR for CA certificates, or TYPE_VOMSDIR for VOMS certificate.
    * `aggressive` – – if true, loading of data will continue even if a particular file could not be loaded, while if false loading will stop as soon as an error occur.

  - **Throws**

    * `java.io.IOException` – if type is neither TYPE_CADIR nor TYPE_VOMSDIR.
    * `java.security.cert.CertificateException` – if there are parsing errors while loading a certificate.
    * `java.security.cert.CRLException` – if there are parsing errors while loading a CRL.

16.5.7 METHODS

- **getAACandidate**

  `public java.security.cert.X509Certificate[]` **getAACandidate**`( javax.security.auth.x500.X500P`
  issuer`, java.lang.String` **voName** `)`

  - **Description**

    Gets an array of candidate issuer certificates for an AC with the given issuer and belonging to the given VO.

  - **Parameters**

    * `issuer` – The issuer of the AC.
    * `voName` – The name of the VO.

  - **Returns** – the array of candidates, or null if none is found.

- **getCAs**

  `public java.util.Hashtable` **getCAs**`( )`

  - **Returns** – hashtable containing CA certificates. The key is the PKIUtils.getHash() of the subject of the CA. The value is a Vector containing all the CA certificates with the given hash.

  - **See also**

    * `PKIUtils.getHash(java.security.cert.X509Certificate)` (in 16.6.5, page 101)
    * `PKIUtils.getHash(javax.security.auth.x500.X500Principal)` (in 16.6.5, page 101)

* `PKIUtils.getHash(org.bouncycastle.jce.X509Principal)` (in 16.6.5, page 102)
* `java.util.Vector`

- **getCRLs**
  `public java.util.Hashtable` **getCRLs**`( )`
  - **Returns** – hashtable containing CRL. The key is the PKIUtils.getHash() of the issuer of the CRL. The value is a Vector containing all the CRL with the given hash.
  - **See also**
    * `PKIUtils.getHash(java.security.cert.X509Certificate)` (in 16.6.5, page 101)
    * `PKIUtils.getHash(javax.security.auth.x500.X500Principal)` (in 16.6.5, page 101)
    * `PKIUtils.getHash(org.bouncycastle.jce.X509Principal)` (in 16.6.5, page 102)
    * `java.util.Vector`
- **getLSC**
  `public LSCFile` **getLSC**`( java.lang.String` **voName**`, java.lang.String` **hostName** `)`
  - **Description**
    Gets the LSC file corresponding to the given VO, for the given server.
  - **Parameters**
    * `voName` – – The name of the VO.
    * `hostName` – – The hostname of the issuing server.
  - **Returns** – The corresponding LSCFile object, or null if none is present.
- **getSignings**
  `public java.util.Hashtable` **getSignings**`( )`
  - **Returns** – hashtable containing SigningPolicy objects. The key is the PKIUtils.getHash() of the issuer of the SigningPolicy. The value is a Vector containing all the CRL with the given hash.
  - **See also**
    * `SigningPolicy` (in 16.8, page 107)
    * `PKIUtils.getHash(java.security.cert.X509Certificate)` (in 16.6.5, page 101)
    * `PKIUtils.getHash(javax.security.auth.x500.X500Principal)` (in 16.6.5, page 101)
    * `PKIUtils.getHash(org.bouncycastle.jce.X509Principal)` (in 16.6.5, page 102)
    * `java.util.Vector`
- **load**
  `public void` **load**`( ) throws java.io.IOException, java.security.cert.CertificateException, java.security.cert.CRLException`
  - **Description**
    Loads the files from the directory specified in the constructors
  - **Throws**
    * `java.io.IOException` – if type is neither TYPE_CADIR nor TYPE_VOMSDIR.

  - `java.security.cert.CertificateException` – if there are parsing errors while loading a certificate.
  - `java.security.cert.CRLException` – if there are parsing errors while loading a CRL.

- **refresh**
  `public synchronized void` **refresh**`( )`
  - **Description**
    Refreshes the content of the PKIStore object.

- **rescheduleRefresh**
  `public void` **rescheduleRefresh**`( int` **millisec** `)`
  - **Description**
    Changes the interval between refreshes of the store.
  - **Parameters**
    - `millisec` – New interval (in milliseconds)

- **setAggressive**
  `public void` **setAggressive**`( boolean` **b** `)`
  - **Description**
    Changes the aggressive mode of the store.
  - **Parameters**
    - `b` – – if true (default) load as much as possible, otherwise stop loading at the first error.

- **stopRefresh**
  `public void` **stopRefresh**`( )`
  - **Description**
    Stop all refreshes. NOTE: This method must ALWAYS be called prior to disposing of a PKIStore object. The penalty for not doing it is a memor leak.

16.6 CLASS PKIUTILS

16.6.1 DECLARATION

public class PKIUtils
extends java.lang.Object

16.6.2 CONSTRUCTOR SUMMARY

PKIUtils()

16.6.3 METHOD SUMMARY

checkIssued(X509Certificate, X509Certificate) Checks if a certificate issued another certificate, according to RFC 3280.
DNCompare(String, String) Compares two DNs for equality, taking into account different representations for the Emaail and UserID tags.
getAKID(X509Certificate) Gets the AuthorityKeyIdentifier extension form the passed certificate.
getBaseName(File) Gets the basename of a file.
getBasicConstraints(X509Certificate) Gets the BasicConstraints extension form the passed certificate.
getHash(byte[]) Gets the MD5 hash value of the given byte array.
getHash(X500Principal) Gets the MD5 hash value of the given principal.
getHash(X509Certificate) Gets the MD5 hash value of the subject of the given certificate.
getHash(X509CRL) Gets the MD5 hash value of the issuer of the given CRL.
getHash(X509Principal) Gets the MD5 hash value of the given principal.
getOpenSSLFormatPrincipal(Principal) Gets an OpenSSL-style representation of a principal.
getSKID(X509Certificate) Gets the SubjectKeyIdentifier extension form the passed certificate.
isCA(X509Certificate) Checks if the passed certificate is a CA certificate.
isProxy(X509Certificate) Checks if the passed certificate is a proxy certificate.
loadCertificates(File) Loads a set of credentials from a file.
loadCertificates(String) Loads a set of credentials from a file.
loadCRL(File) Loads a CRL from a file.
loadCRL(String) Loads a CRL from a file.
readObject(File) Reads either a certificate or a CRL from a file.
selfIssued(X509Certificate) Checks if the give certificate is self-issued.
skipToCertBeginning(BufferedInputStream) Prepares a BufferedInputStream to read either a certificate or a CRL from it.

16.6.4 CONSTRUCTORS

- **PKIUtils**
  `public` **PKIUtils**`( )`

16.6.5 METHODS

- **checkIssued**
  `public static boolean` **checkIssued**`( java.security.cert.X509Certificate` **issuer**`, java.security` **issued** `)`
  - **Description**
    Checks if a certificate issued another certificate, according to RFC 3280.
  - **Parameters**
    * `issuer` – The candidate issuer certificate.
    * `issued` – The candidate issued certificate.

  - **Returns** – true if issuer issued issued, false othersie.

- **DNCompare**
  `public static boolean` **DNCompare**`( java.lang.String` **dn1**`, java.lang.String` **dn2** `)`

  - **Description**
    Compares two DNs for equality, taking into account different representations for the Emaail and UserID tags.
  - **Parameters**
    * `dn1` – the first dn to compare.
    * `dn2` – the second dn to compare
  - **Returns** – true if dn1 and dn2 are equal, false otherwise.

- **getAKID**
  `public static org.bouncycastle.asn1.x509.AuthorityKeyIdentifier` **getAKID**`( java.security.cer` **cert** `)`

  - **Description**
    Gets the AuthorityKeyIdentifier extension form the passed certificate.
  - **Parameters**
    * `cert` – The certificate from which to get the extension.
  - **Returns** – the extension if present, or null if not present.

- **getBaseName**
  `public static java.lang.String` **getBaseName**`( java.io.File` **f** `)`

  - **Description**
    Gets the basename of a file.
  - **Parameters**
    * `f` – File object representing a file.
  - **Returns** – a string representing the file name, minus the path.

- **getBasicConstraints**
  `public static org.bouncycastle.asn1.x509.BasicConstraints` **getBasicConstraints**`( java.securit` **cert** `)`

  - **Description**
    Gets the BasicConstraints extension form the passed certificate.
  - **Parameters**
    * `cert` – The certificate from which to get the extension.
  - **Returns** – the extension if present, or null if not present.

- **getHash**
  `public static java.lang.String` **getHash**`( byte[]` **name** `)`

  - **Description**
    Gets the MD5 hash value of the given byte array.

  - **Parameters**
    * `name` – the data from which to compute the hash.
  - **Returns** – the hash value.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if crl is null.
    * `InvalidStateException` – if the MD5 algorithm is not supported.

- **getHash**
  `public static java.lang.String` **getHash**`( javax.security.auth.x500.X500Principal` **principal** `)`
  - **Description**
    Gets the MD5 hash value of the given principal.
  - **Parameters**
    * `principal` – the principal.
  - **Returns** – the hash value.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if crl is null.
    * `InvalidStateException` – if the MD5 algorithm is not supported.

- **getHash**
  `public static java.lang.String` **getHash**`( java.security.cert.X509Certificate` **x509** `)`
  - **Description**
    Gets the MD5 hash value of the subject of the given certificate.
  - **Parameters**
    * `x509` – The certificate from which to get the subject.
  - **Returns** – the hash value.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if x509 is null.
    * `InvalidStateException` – if the MD5 algorithm is not supported.

- **getHash**
  `public static java.lang.String` **getHash**`( java.security.cert.X509CRL` **crl** `)`
  - **Description**
    Gets the MD5 hash value of the issuer of the given CRL.
  - **Parameters**
    * `crl` – The CRL from which to get the issuer.
  - **Returns** – the hash value.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if crl is null.
    * `InvalidStateException` – if the MD5 algorithm is not supported.

- **getHash**
  `public static java.lang.String` **getHash**`( org.bouncycastle.jce.X509Principal` **principal** `)`
  - **Description**
    Gets the MD5 hash value of the given principal.
  - **Parameters**
    * `principal` – the principal.
  - **Returns** – the hash value.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if crl is null.
    * `InvalidStateException` – if the MD5 algorithm is not supported.
- **getOpenSSLFormatPrincipal**
  `public static java.lang.String` **getOpenSSLFormatPrincipal**`( java.security.Principal` **principal** `)`
  - **Description**
    Gets an OpenSSL-style representation of a principal.
  - **Parameters**
    * `principal` – the principal
  - **Returns** – a String representing the principal.
- **getSKID**
  `public static org.bouncycastle.asn1.x509.SubjectKeyIdentifier` **getSKID**`( java.security.cert.` **cert** `)`
  - **Description**
    Gets the SubjectKeyIdentifier extension form the passed certificate.
  - **Parameters**
    * `cert` – The certificate from which to get the extension.
  - **Returns** – the extension if present, or null if not present.
- **isCA**
  `public static boolean` **isCA**`( java.security.cert.X509Certificate` **cert** `)`
  - **Description**
    Checks if the passed certificate is a CA certificate.
  - **Parameters**
    * `cert` – the candidate CA certificate.
  - **Returns** – true if cert is a CA certificate.
- **isProxy**
  `public static boolean` **isProxy**`( java.security.cert.X509Certificate` **cert** `)`
  - **Description**
    Checks if the passed certificate is a proxy certificate. Recognizes GT2, GT3 and GT4 proxies.

  - **Parameters**
    * `cert` – the candidate proxy certificate.
  - **Returns** – true if cert is a proxy certificate.

- **loadCertificates**
  `public static java.security.cert.X509Certificate[]` **loadCertificates**`( java.io.File` **file** `) throws java.security.cert.CertificateException`
  - **Description**
    Loads a set of credentials from a file.
  - **Parameters**
    * `file` – the File object from which to load the certificates.
  - **Returns** – an array containing the certificates that were present in the file.
  - **Throws**
    * `java.security.cert.CertificateException` – if there were problems parsing the certificates.
    * `java.lang.IllegalArgumentException` – if the file cannot be found.
  - **See also**
    * `java.io.File`

- **loadCertificates**
  `public static java.security.cert.X509Certificate[]` **loadCertificates**`( java.lang.String` **filename** `) throws java.security.cert.CertificateException`
  - **Description**
    Loads a set of credentials from a file.
  - **Parameters**
    * `filename` – the name of the file from which to load the certificates.
  - **Returns** – an array containing the certificates that were present in the file.
  - **Throws**
    * `java.security.cert.CertificateException` – if there were problems parsing the certificates.
    * `java.lang.IllegalArgumentException` – if the file cannot be found.

- **loadCRL**
  `public static java.security.cert.X509CRL` **loadCRL**`( java.io.File` **file** `) throws java.security.`
  - **Description**
    Loads a CRL from a file.
  - **Parameters**
    * `file` – the File object from which to load the CRL.
  - **Returns** – an array containing the certificates that were present in the file.
  - **Throws**
    * `java.security.cert.CRLException` – if there were problems parsing the CRL.
    * `java.lang.IllegalArgumentException` – if the file cannot be found.

- **loadCRL**
  `public static java.security.cert.X509CRL` **loadCRL**`( java.lang.String` **filename** `) throws java.security.cert.CRLException`
  - **Description**
    Loads a CRL from a file.
  - **Parameters**
    * `filename` – the name of the file from which to load the CRL.
  - **Returns** – an array containing the certificates that were present in the file.
  - **Throws**
    * `java.security.cert.CRLException` – if there were problems parsing the CRL.
    * `java.lang.IllegalArgumentException` – if the file cannot be found.
- **readObject**
  `public static java.lang.Object` **readObject**`( java.io.File` **f** `) throws java.io.IOException, java.security.cert.CertificateException, java.security.cert.CRLException`
  - **Description**
    Reads either a certificate or a CRL from a file.
  - **Parameters**
    * `f` – the file from which to read;
  - **Returns** – the Object loaded.
  - **Throws**
    * `java.io.IOException` – if there have been problems reading the file.
    * `java.security.cert.CertificateException` – if there have been problems parsing the certificate.
    * `java.security.cert.CRLException` – if there have been problems parsing the CRL.
- **selfIssued**
  `public static boolean` **selfIssued**`( java.security.cert.X509Certificate` **cert** `)`
  - **Description**
    Checks if the give certificate is self-issued.
  - **Parameters**
    * `cert` – The certificate to check.
  - **Returns** – true if the certificate is self-issued, false otherwise.
- **skipToCertBeginning**
  `public static int` **skipToCertBeginning**`( java.io.BufferedInputStream` **stream** `) throws java.io.IOException`
  - **Description**
    Prepares a BufferedInputStream to read either a certificate or a CRL from it. Skips everything in front of "—–BEGIN" in the stream.
  - **Parameters**
    * `stream` – The stream to read and skip.

- **Returns** – CERT if a certificate is the next object to be read from the stream, CRL if the next object is a CRL, -1 if the next object is of type unknown.
- **Throws**
  - `java.io.IOException` – Thrown if there is a problem skipping. Note: this a modified version of code originally written by Joni Hakhala

16.7 CLASS PKIVERIFIER

16.7.1 DECLARATION

public class PKIVerifier
extends java.lang.Object

16.7.2 FIELD SUMMARY

AUTHORITY_KEY_IDENTIFIER
BASIC_CONSTRAINTS_IDENTIFIER
KEY_USAGE_IDENTIFIER
PROXYCERTINFO
PROXYCERTINFO_OLD
SUBJECT_KEY_IDENTIFIER
TARGET

16.7.3 CONSTRUCTOR SUMMARY

PKIVerifier() Initializes the verifier.
PKIVerifier(VOMSTrustStore) Initializes the verifier.
PKIVerifier(VOMSTrustStore, PKIStore) Initializes the verifier.

16.7.4 METHOD SUMMARY

cleanup() Cleans up resources allocated by the verifier.
setCAStore(PKIStore) Sets a new CAStore.
setVOMSStore(VOMSTrustStore) Sets a new VOMSStore.
verify(AttributeCertificate) Verifies an Attribute Certificate according to RFC 3281.
verify(X509Certificate[]) Verifies an certificate chain according to RFC 3280.

16.7.5 FIELDS

- public static final java.lang.String **SUBJECT_KEY_IDENTIFIER**
- public static final java.lang.String **AUTHORITY_KEY_IDENTIFIER**
- public static final java.lang.String **PROXYCERTINFO**
- public static final java.lang.String **PROXYCERTINFO_OLD**

- public static final java.lang.String **BASIC_CONSTRAINTS_IDENTIFIER**
- public static final java.lang.String **KEY_USAGE_IDENTIFIER**
- public static final java.lang.String **TARGET**

16.7.6 CONSTRUCTORS

- **PKIVerifier**
  `public` **PKIVerifier**`( ) throws java.io.IOException, java.security.cert.CertificateException, java.security.cert.CRLException`
  - **Description**
    Initializes the verifier. The CA store is initialized at: "/etc/grid-security/certificates.", while the VOMS store is initialized at "/etc/grid-security/vomsdir".
  - **Throws**
    * `java.io.IOException` – if there have been IO errors.
    * `java.security.cert.CertificateException` – if there have been problems parsing a certificate
    * `java.security.cert.CRLException` – if there have been problems parsing a CRL.
- **PKIVerifier**
  `public` **PKIVerifier**`( ac.VOMSTrustStore` **vomsStore** `) throws java.io.IOException, java.security.cert.CertificateException, java.security.cert.CRLException`
  - **Description**
    Initializes the verifier. The CA store is initialized at: "/etc/grid-security/certificates."
  - **Parameters**
    * `vomsStore` – the VOMSTrustStore object which represents the vomsdir store.
  - **Throws**
    * `java.io.IOException` – if there have been IO errors.
    * `java.security.cert.CertificateException` – if there have been problems parsing a certificate
    * `java.security.cert.CRLException` – if there have been problems parsing a CRL.
- **PKIVerifier**
  `public` **PKIVerifier**`( ac.VOMSTrustStore` **vomsStore**`, PKIStore` **caStore** `)`
  - **Description**
    Initializes the verifier.
  - **Parameters**
    * `vomsStore` – the VOMSTrustStore object which represents the vomsdir store.
    * `caStore` – the PKIStore object which represents the CA store.

16.7.7 METHODS

- **cleanup**
  `public void` **cleanup**`( )`
  - **Description**
    Cleans up resources allocated by the verifier. This method MUST be called prior to disposal of this object, otherwise memory leaks and runaway threads will occur.
- **setCAStore**
  `public void` **setCAStore**`( PKIStore` **store** `)`
  - **Description**
    Sets a new CAStore.
  - **Parameters**
    * `store` – the new CA store.
- **setVOMSStore**
  `public void` **setVOMSStore**`( ac.VOMSTrustStore` **store** `)`
  - **Description**
    Sets a new VOMSStore.
  - **Parameters**
    * `store` – the new VOMS store.
- **verify**
  `public boolean` **verify**`( ac.AttributeCertificate` **ac** `)`
  - **Description**
    Verifies an Attribute Certificate according to RFC 3281.
  - **Parameters**
    * `ac` – the Attribute Certificate to verify.
  - **Returns** – true if the attribute certificate is verified, false otherwise.
- **verify**
  `public boolean` **verify**`( java.security.cert.X509Certificate[]` **certs** `)`
  - **Description**
    Verifies an certificate chain according to RFC 3280.
  - **Parameters**
    * `certs` – the chain to verify.
  - **Returns** – true if the chain is verified, false otherwise.

16.8 CLASS SIGNINGPOLICY

The purpose of this class is to represent a *.signing_policy file.

16.8.1 DECLARATION

public class SigningPolicy
extends java.lang.Object

16.8.2 CONSTRUCTOR SUMMARY

SigningPolicy(File) Loads a *.signing_policy file.

16.8.3 METHOD SUMMARY

findIssuer(String) Finds the record in the signing policy which deals with the specified issuer.
findIssuer(String, int) Finds the record in the signing policy which deals with the specified issuer, starting from a specified record.
getAccessIDCA() Gets the AccessIDCA from the current record.
getCondSubjects() Gets the CondSubjects from the current record.
getName() Gets the basename of the file from which this was loaded.
getPosRights() Gets the PosRights from the current record.
setCurrent(int) Sets the indicate record as the current record.

16.8.4 CONSTRUCTORS

- **SigningPolicy**
  `public` **SigningPolicy**`( java.io.File` **f** `) throws java.io.IOException`
  - **Description**
    Loads a *.signing_policy file.
  - **Parameters**
    * `f` – the File from which to load the Signing Policy.
  - **Throws**
    * `java.io.IOException` – if there have been problems loading the file.

16.8.5 METHODS

- **findIssuer**
  `public int` **findIssuer**`( java.lang.String` **issuer** `)`
  - **Description**
    Finds the record in the signing policy which deals with the specified issuer.
  - **Parameters**
    * `issuer` – an OpenSSL-style representation of the issuer.
  - **Returns** – the record number, or -1 if none is found.
- **findIssuer**
  `public int` **findIssuer**`( java.lang.String` **issuer**`, int` **previous** `)`

  - **Description**
    Finds the record in the signing policy which deals with the specified issuer, starting from a specified record.
  - **Parameters**
    * `issuer` – an OpenSSL-style representation of the issuer.
    * `previous` – the previous match, or -1 if ther was no previous match.
  - **Returns** – the record number, or -1 if none is found.

- **getAccessIDCA**
  `public java.lang.String` **getAccessIDCA**`( )`
  - **Description**
    Gets the AccessIDCA from the current record.
  - **Returns** – the AccessIDCA.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if the record number has not been set.

- **getCondSubjects**
  `public java.util.Vector` **getCondSubjects**`( )`
  - **Description**
    Gets the CondSubjects from the current record.
  - **Returns** – a Vector of CondSubjects. Each element is a String.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if the record number has not been set.

- **getName**
  `public java.lang.String` **getName**`( )`
  - **Description**
    Gets the basename of the file from which this was loaded.
  - **Returns** – the basename or null if nothign was loaded.

- **getPosRights**
  `public java.lang.String` **getPosRights**`( )`
  - **Description**
    Gets the PosRights from the current record.
  - **Returns** – the PosRight
  - **Throws**
    * `java.lang.IllegalArgumentException` – if the record number has not been set.

- **setCurrent**
  `public void` **setCurrent**`( int` **index** `)`
  - **Description**
    Sets the indicate record as the current record.
  - **Parameters**
    * `index` – the record number
  - **Throws**
    * `java.lang.IllegalArgumentException` – if the record number is too great or 0.

16.9 CLASS VOMSATTRIBUTE

Representation of the authorization information (VO, server address and list of Fully Qualified Attribute Names, or FQANs) contained in a VOMS attribute certificate.

16.9.1 DECLARATION

public class VOMSAttribute
extends java.lang.Object

16.9.2 CONSTRUCTOR SUMMARY

VOMSAttribute(AttributeCertificate) Parses the contents of an attribute certificate.
NOTE: Cryptographic signatures, time stamps etc.

16.9.3 METHOD SUMMARY

getAC() Deprecated! Use the getXXX() methods instead.
getCertList() Gets the certificates that signed the AC, if the ACCerts extension is present.
getFullAttributes() Gets a copy of the Generic Attributes extension.
getFullyQualifiedAttributes()
getHolder() Returns an String representation of the AC holder.
getHolderX509() Returns an OpenSSL-style representation of the AC holder.
getHost() Returns the hostname of the issuing VOMS server.
getHostPort() Returns the address of the issuing VOMS server, on the form `<host>:<port>`
getIssuer() Returns an OpenSSL-style representation of the AC issuer.
getIssuerX509() Returns an OpenSSL-style representation of the AC issuer.
getListOfFQAN()
getNotAfter() Returns the end date of the AC validity.
getNotBefore() Return the start date of the AC validity.
getPort() Returns the port on which the issuing VOMS server is listening
getSerial() Returns the serial number of the AC.
getSignature() Returns the signature of the AC.
getTargets() Gets the targets of this AC.
getVO() Returns the VO name
isHolder(X509Certificate) Checks the given X509 certificate to see if it is the holder of the AC.
isIssuer(X509Certificate) Checks the given X509 certificate to see if it is the issuer of the AC.
isValid() Checks if the Attribute is valid.
toString() Gets a (brief) string representation of this attribute.
validAt(Date) Checks if the AC was valid at the provided timestamp.

16.9.4 CONSTRUCTORS

- **VOMSAttribute**
  `public` **VOMSAttribute**`( ac.AttributeCertificate` **ac** `)`

– **Description**

Parses the contents of an attribute certificate.
NOTE: Cryptographic signatures, time stamps etc. will **not** be checked.

– **Parameters**

* `ac` – the attribute certificate to parse for VOMS attributes

16.9.5 METHODS

- **getAC**
`public ac.AttributeCertificate` **getAC**`( )`

Deprecated! Direct access to the Attribute Certificate is going to be removed. Use the getXXX methods in this same classe instead.

– **Returns** – The AttributeCertificate containing the VOMS information

- **getCertList**
`public ac.ACCerts` **getCertList**`( )`

– **Description**

Gets the certificates that signed the AC, if the ACCerts extension is present.

– **Returns** – the ACCerts extension, or null if it is not present.

– **Throws**

* `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getFullAttributes**
`public ac.FullAttributes` **getFullAttributes**`( )`

– **Description**

Gets a copy of the Generic Attributes extension.

– **Returns** – the attributes, or null if they are not present.

– **Throws**

* `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getFullyQualifiedAttributes**
`public java.util.List` **getFullyQualifiedAttributes**`( )`

– **Returns** – List of String of the VOMS fully qualified attributes names (FQANs):
`vo[/group[/group2...]][/Role=[role]][/Capability=capability]`

– **Throws**

* `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getHolder**
`public java.lang.String` **getHolder**`( )`

– **Description**

Returns an String representation of the AC holder.

– **Returns** – the AC holder.

  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getHolderX509**
  `public java.lang.String` **getHolderX509**`( )`
  - **Description**
    Returns an OpenSSL-style representation of the AC holder.
  - **Returns** – the AC holder.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getHost**
  `public java.lang.String` **getHost**`( )`
  - **Description**
    Returns the hostname of the issuing VOMS server.
  - **Returns** – hostname.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getHostPort**
  `public java.lang.String` **getHostPort**`( )`
  - **Description**
    Returns the address of the issuing VOMS server, on the form `<host>:<port>`
  - **Returns** – String
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getIssuer**
  `public java.lang.String` **getIssuer**`( )`
  - **Description**
    Returns an OpenSSL-style representation of the AC issuer.
  - **Returns** – the AC issuer.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getIssuerX509**
  `public java.lang.String` **getIssuerX509**`( )`
  - **Description**
    Returns an OpenSSL-style representation of the AC issuer.
  - **Returns** – the AC issuer.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getListOfFQAN**
  `public java.util.List` **getListOfFQAN**`( )`
  - **Returns** – List of FQAN of the VOMS fully qualified attributes names (FQANs)
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.
  - **See also**
    * `FQAN` (in 16.2, page 91)
- **getNotAfter**
  `public java.util.Date` **getNotAfter**`( ) throws java.text.ParseException`
  - **Description**
    Returns the end date of the AC validity.
  - **Returns** – the end Date.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.
- **getNotBefore**
  `public java.util.Date` **getNotBefore**`( ) throws java.text.ParseException`
  - **Description**
    Return the start date of the AC validity.
  - **Returns** – the start Date.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.
- **getPort**
  `public int` **getPort**`( )`
  - **Description**
    Returns the port on which the issuing VOMS server is listening
  - **Returns** – the port, or -1 if the informations could not be found.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.
- **getSerial**
  `public java.lang.String` **getSerial**`( )`
  - **Description**
    Returns the serial number of the AC.
  - **Returns** – the serial number of the AC.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.
- **getSignature**
  `public byte[]` **getSignature**`( )`

  - **Description**
    Returns the signature of the AC.
  - **Returns** – the byte representation of the AC signature.

- **getTargets**
  `public ac.ACTargets` **getTargets**`( )`
  - **Description**
    Gets the targets of this AC.
  - **Returns** – the ACTargets extension if present, or null otherwise.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **getVO**
  `public java.lang.String` **getVO**`( )`
  - **Description**
    Returns the VO name
  - **Returns** – the VO name
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **isHolder**
  `public boolean` **isHolder**`( java.security.cert.X509Certificate` **cert** `)`
  - **Description**
    Checks the given X509 certificate to see if it is the holder of the AC.
  - **Parameters**
    * `cert` – the X509 certificate to check.
  - **Returns** – true if the give certificate is the holder of the AC.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **isIssuer**
  `public boolean` **isIssuer**`( java.security.cert.X509Certificate` **cert** `)`
  - **Description**
    Checks the given X509 certificate to see if it is the issuer of the AC.
  - **Parameters**
    * `cert` – the X509 certificate to check.
  - **Returns** – true if the give certificate is the issuer of the AC.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **isValid**
  `public boolean` **isValid**`( )`

  - **Description**
    Checks if the Attribute is valid. Only checks start and end of validity.
  - **Returns** – true if is valid, false otherwise.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded.

- **toString**
  `public java.lang.String` **toString**`( )`
  - **Description**
    Gets a (brief) string representation of this attribute.
  - **Returns** – the Representation.

- **validAt**
  `public boolean` **validAt**`( java.util.Date` **date** `)`
  - **Description**
    Checks if the AC was valid at the provided timestamp.
  - **Parameters**
    * `date` – if `null`, current time is used
  - **Returns** – true if the AC was valid at the time in question.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if no Attribute Certificate has been loaded or the dates have been encoded incorrectly.

16.10 CLASS VOMSTRUSTMANAGER

This is a TrustManager subclass to handle proxies in SSL connections.

16.10.1 DECLARATION

public class VOMSTrustManager
extends javax.net.ssl.X509TrustManager

16.10.2 CONSTRUCTOR SUMMARY

VOMSTrustManager(String) Create an object reading the credentials from a directory.
VOMSTrustManager(PKIStore) Create an object from an existing Store.

16.10.3 METHOD SUMMARY

stop() Does cleanup before object disposal.
checkClientTrusted(X509Certificate[], String) See X509TrustManager for documentation.
checkServerTrusted(X509Certificate[], String) See X509TrustManager for documentation.
getAcceptedIssuers() See X509TrustManager for Documentation .

16.10.4 CONSTRUCTORS

- **VOMSTrustManager**
  `public` **VOMSTrustManager**`(String` **dir**`)`
  - **Description**
    Creates a new trust manager loading the credentials from a directory
  - **Parameters**
    * `dir` – The directory which contains the certificates
- **VOMSTrustManager**
  `public` **VOMSTrustManager**`(PKIStore` **store**`)`
  - **Description**
    Creates a new trust manger loading the credentials from an existing store
  - **Parameters**
    * `store` – The existing store

16.10.5 METHODS

- **stop**
  stop()
  - **Description** Cleans up an object prior to disposal. IMPORTANT: This method MUST be called before disposing of the reference, on pain of a memory leak.

16.11 CLASS VOMSKEYMANAGER

The class will implement the Key Manager for setting up SSL connections capable of handling proxies.

16.11.1 DECLARATION

public class VOMSKeyManager
extends javax.net.ssl.X509KeyManager

16.11.2 FIELD SUMMARY

TYPE_PKCS12 A certificate in PKCS12 format.
TYPE_PEM A certificate in PEM format.

16.11.3 CONSTRUCTOR SUMMARY

VOMSKeyManager(String, String, String) Create an object specifying certificate, key and password.
VOMSKeyManager(String, String, String, int) Create an object specifying certificate, key, password and type.

16.11.4 METHOD SUMMARY

chooseClientAlias(String[], Principal[], Socket) See X509KeyManager documentation.
chooseServerAlias(String, Principal[], Socket) See X509KeyManager documentation.
getCertificateChain(String) See X509KeyManager documentation.
getClientAliases(String, Principal[]) See X509KeyManager documentation.
getServerAliases(String, Principal[]) See X509KeyManager documentation.
getPrivateKey(String) See X509KeyManager documentation.

16.11.5 FIELDS

- public static final int **TYPE_PKCS12**
  - This Certificate is in PKCS12 format
- public static final int **TYPE_PEM**
  - This certificate is in PEM format

16.11.6 CONSTRUCTORS

- **VOMSKeyManager**
  `public` **VOMSKeyManager**`(String` **cert**`, String` **key**`, String` **password**`)`
  - **Description**
    Initializes a KeyManager object starting from a Certificate, a Key and a Password
  - **Parameters**
    * `cert` The certificate
    * `key` The key
    * `password` The password
- **VOMSKeyManager**
  `public` **VOMSKeyManager**`(String` **cert**`, String` **key**`, String` **password**`, int` **type**`)`
  - **Description**
    Initializes a KeyManager object starting from a Certificate, a Key and a Password
  - **Parameters**
    * `cert` The certificate
    * `key` The key
    * `password` The password
    * `type` The type, either TYPE_PKCS12 or TYPE_PEM

16.12 CLASS VOMSVALIDATOR

The main (top) class to use for extracting VOMS information from a certificate and/or certificate chain. The VOMS information can simply be parsed or validated. No validation is performed on the certificate chain – that is assumed to already have happened.

The certificate chain is assumed to already be validated. It is also assumed to be sorted in TLS order, that is certificate issued by trust anchor first and client certificate last.
Example of use: this will validate any VOMS attributes in the certificate chain and check if any of the attributes grants the user the "admin" role in the group (VO) "MyVO".

```
boolean isAdmin = new VOMSValidator(certChain).validate().getRoles("MyVO").contains("admin");
```

16.12.1 DECLARATION

public class VOMSValidator
extends java.lang.Object

16.12.2 FIELD SUMMARY

isParsed
isValidated
myFQANTree
myValidatedChain
myValidator
myVomsAttributes
theTrustStore
VOMS_EXT_OID
vomsStore

16.12.3 CONSTRUCTOR SUMMARY

VOMSValidator(X509Certificate) Convenience constructor in the case where you have a single cert and not a chain.
VOMSValidator(X509Certificate[]) Convenience constructor
Same as `VOMSValidator(validatedChain, null)`
VOMSValidator(X509Certificate[], ACValidator) If `validatedChain` is `null`, a call to `setValidatedChain()` MUST be made before calling `parse()` or `validate()`.

16.12.4 METHOD SUMMARY

cleanup() Cleans up the object.
getCapabilities(String) Returns a list of all capabilities attributed to a (sub)group, by combining all VOMS attributes in a hiearchial fashion.
getRoles(String) Returns a list of all roles attributed to a (sub)group, by combining all VOMS attributes in a hiearchial fashion.
getVOMSAttributes() Returns a list of VOMS attributes, parsed and possibly validated.
isValidated()
parse() Parses the assumed-validated certificate chain (which may also include proxy certs) for any occurances of VOMS extensions containing attribute certificates issued to the end entity in the certificate chain.

parse(X509Certificate[]) Parses the assumed-validated certificate chain (which may also include proxy certs) for any occurances of VOMS extensions containing attribute certificates issued to the end entity in the certificate chain.

setClientChain(X509Certificate[]) Convenience method: enables you to reuse a `VOMSValidator` instance for another client chain, thus avoiding overhead in instantiating validators and trust stores and other potentially expensive operations.

setTrustStore(ACTrustStore) Sets the ACTrustStore instance to use with the default ACValidator.

setTrustStore(VOMSTrustStore) Sets the trustStore to use with the default ACValidator.

toString()

validate() Parses the assumed-validated certificate chain (which may also include proxy certs) for any occurances of VOMS extensions containing attribute certificates issued to the end entity in the certificate chain.

16.12.5 FIELDS

- public static final java.lang.String **VOMS_EXT_OID**
- protected static ac.ACTrustStore **theTrustStore**
- protected ac.ACValidator **myValidator**
- protected java.security.cert.X509Certificate **myValidatedChain**
- protected java.util.Vector **myVomsAttributes**
- protected boolean **isParsed**
- protected boolean **isValidated**
- protected VOMSValidator.FQANTree **myFQANTree**
- protected static ac.VOMSTrustStore **vomsStore**

16.12.6 CONSTRUCTORS

- **VOMSValidator**
  `public` **VOMSValidator**`( java.security.cert.X509Certificate` **validatedCert** `)`
  - **Description**
    Convenience constructor in the case where you have a single cert and not a chain.
  - **Parameters**
    * `validatedCert` –
  - **See also**
    * `VOMSValidator(java.security.cert.X509Certificate[])` (in 16.12.6, page 119)
- **VOMSValidator**
  `public` **VOMSValidator**`( java.security.cert.X509Certificate[]` **validatedChain** `)`

- **Description**

  Convenience constructor
  Same as `VOMSValidator(validatedChain, null)`

- **Parameters**

  * `validatedChain` –

- **VOMSValidator**
  `public` **VOMSValidator**`( java.security.cert.X509Certificate[]` **validatedChain**`, ac.ACValidator` **acValidator** `)`

  - **Description**

    If `validatedChain` is `null`, a call to `setValidatedChain()` MUST be made before calling `parse()` or `validate()`.

  - **Parameters**

    * `validatedChain` – The (full), validated certificate chain
    * `acValidator` – The AC validator implementation to use (null is default with a BasicVOMSTrustStore)

  - **See also**

    * `ac.ACValidator` (in 16.20, page 136)
    * `BasicVOMSTrustStore` (in 16.1, page 89)

16.12.7 METHODS

- **cleanup**
  `public void` **cleanup**`( )`

  - **Description**

    Cleans up the object. This method MUST be called before disposing of the object, on pains of a memory leak.

- **getCapabilities**
  `public java.util.List` **getCapabilities**`( java.lang.String` **subGroup** `)`

  Deprecated! Capabilities are deprecated.

  - **Description**

    Returns a list of all capabilities attributed to a (sub)group, by combining all VOMS attributes in a hiearchial fashion.
    Note: One of the methods `parse()` or `validate()` must have been called before calling this method. Otherwise, an `IllegalStateException` is thrown.

  - **Parameters**

    * `subGroup` –

  - **Returns** – A list containing all the capabilities

  - **See also**

    * `VOMSValidator.FQANTree` (in 16.13, page 123)

- **getRoles**
  `public java.util.List` **getRoles**`( java.lang.String` **subGroup** `)`
  - **Description**
    Returns a list of all roles attributed to a (sub)group, by combining all VOMS attributes in a hiearchial fashion.
    Note: One of the methods `parse()` or `validate()` must have been called before calling this method. Otherwise, an `IllegalStateException` is thrown.
  - **Parameters**
    * `subGroup` –
  - **Returns** – the List of roles.
  - **See also**
    * `VOMSValidator.FQANTree` (in 16.13, page 123)

- **getVOMSAttributes**
  `public java.util.List` **getVOMSAttributes**`( )`
  - **Description**
    Returns a list of VOMS attributes, parsed and possibly validated.
  - **Returns** – List of `VOMSAttribute`
  - **See also**
    * `VOMSAttribute` (in 16.9, page 110)
    * `VOMSValidator.parse()` (in 16.12.7, page 121)
    * `VOMSValidator.validate()` (in 16.12.7, page 123)
    * `VOMSValidator.isValidated()` (in 16.12.7, page 121)

- **isValidated**
  `public boolean` **isValidated**`( )`
  - **Returns** – whether the VOMS attributes are validated or not
  - **See also**
    * `VOMSValidator.validate()` (in 16.12.7, page 123)

- **parse**
  `public VOMSValidator` **parse**`( )`
  - **Description**
    Parses the assumed-validated certificate chain (which may also include proxy certs) for any occurances of VOMS extensions containing attribute certificates issued to the end entity in the certificate chain.
    No validation of timestamps and/or signatures are performed by this method.
    This method returns the object itself, to allow for chaining of commands:
    `new VOMSValidator(certChain).parse().getVOMSAttributes();`
  - **Returns** – the object itself
  - **See also**
    * `VOMSValidator.validate()` (in 16.12.7, page 123)

- **parse**
  `public static java.util.Vector` **parse**`( java.security.cert.X509Certificate[]` **myValidatedChain** `)`

  – **Description**

  Parses the assumed-validated certificate chain (which may also include proxy certs) for any occurances of VOMS extensions containing attribute certificates issued to the end entity in the certificate chain.
  No validation of timestamps and/or signatures are performed by this method.

  – **Returns** – the voms attributes

  – **See also**

  * `VOMSValidator.validate()` (in 16.12.7, page 123)

- **setClientChain**
  `public VOMSValidator` **setClientChain**`( java.security.cert.X509Certificate[]` **validatedChain** `)`

  – **Description**

  Convenience method: enables you to reuse a `VOMSValidator` instance for another client chain, thus avoiding overhead in instantiating validators and trust stores and other potentially expensive operations.
  This method returns the object itself, to allow for chaining of commands:
  `vomsValidator.setValidatedChain(chain).validate().getVOMSAttributes();`

  – **Parameters**

  * `validatedChain` – The new validated certificate chain to inspect

  – **Returns** – the object itself

- **setTrustStore**
  `public static void` **setTrustStore**`( ac.ACTrustStore` **trustStore** `)`

  Deprecated! Use setTrustStore(VOMSTrustStore trustStore) instead.

  – **Description**

  Sets the ACTrustStore instance to use with the default ACValidator. Default is `BasicVOMSTrustStore`

  – **Parameters**

  * `trustStore` –

  – **See also**

  * `VOMSValidator.setTrustStore(ac.VOMSTrustStore)` (in 16.12.7, page 122)
  * `BasicVOMSTrustStore` (in 16.1, page 89)

- **setTrustStore**
  `public static void` **setTrustStore**`( ac.VOMSTrustStore` **trustStore** `)`

  – **Description**

  Sets the trustStore to use with the default ACValidator.

  – **Parameters**

* `trustStore` – the trustStore.

– **See also**

* `ac.VOMSTrustStore` (in 16.15, page 126)

- **toString**
  `public java.lang.String` **toString**`( )`
- **validate**
  `public VOMSValidator` **validate**`( )`

  – **Description**
  Parses the assumed-validated certificate chain (which may also include proxy certs) for any occurances of VOMS extensions containing attribute certificates issued to the end entity in the certificate chain. The attribute certificates are validated: any non-valid entries will be ignored. This method returns the object itself, to allow for chaining of commands:
  `new VOMSValidator(certChain).parse().getVOMSAttributes();`

  – **Returns** – the object itself

  – **See also**

  * `VOMSValidator.parse()` (in 16.12.7, page 121)

16.13 CLASS VOMSVALIDATOR.FQANTREE

Class to sort out the hierarchial properties of FQANs. For example, given the FQANs `/VO/Role=admin` and /VO/SubGroup/Role=user, this means that the applicable roles for /VO/SubGroup is both `admin` as well as `user`

16.13.1 DECLARATION

public class VOMSValidator.FQANTree
extends java.lang.Object

16.13.2 CONSTRUCTOR SUMMARY

VOMSValidator.FQANTree()

16.13.3 METHOD SUMMARY

add(FQAN)
add(List)
getCapabilities(String)
getRoles(String)
traverse(String)

16.13.4 CONSTRUCTORS

- **VOMSValidator.FQANTree**
  `public` **VOMSValidator.FQANTree**`( )`

16.13.5 METHODS

- **add**
  `public void` **add**`( FQAN` **fqan** `)`
- **add**
  `public void` **add**`( java.util.List` **fqans** `)`
- **getCapabilities**
  `public java.util.List` **getCapabilities**`( java.lang.String` **voGroup** `)`
- **getRoles**
  `public java.util.List` **getRoles**`( java.lang.String` **voGroup** `)`
- **traverse**
  `protected VOMSValidator.RoleCaps` **traverse**`( java.lang.String` **voGroup** `)`

Package org.glite.voms.ac

Package Contents *Page*

Interfaces

Classes


```
AttributeCertificateInfo ::= SEQUENCE {
     version              AttCertVersion -- version is v2,
     holder               Holder,
     issuer               AttCertIssuer,
     signature            AlgorithmIdentifier,
     serialNumber         CertificateSerialNumber,
     attrCertValidityPeriod   AttCertValidityPeriod,
     attributes           SEQUENCE OF Attribute,
     issuerUniqueID       UniqueIdentifier OPTIONAL,
     extensions           Extensions OPTIONAL
}

AttCertVersion ::= INTEGER { v2(1) }
```

Validator class capable of validating an Attribute Certificate and verify its signature against a trust store of Attribute Authority certificates.

16.14 INTERFACE ACTRUSTSTORE

NOTE: This interface is deprecated. Use VOMSTrustStore instead.

16.14.1 DECLARATION

public interface ACTrustStore

16.14.2 ALL KNOWN SUBINTERFACES

BasicVOMSTrustStore (in 16.1, page 89)

16.14.3 ALL CLASSES KNOWN TO IMPLEMENT INTERFACE

BasicVOMSTrustStore (in 16.1, page 89)

16.14.4 METHOD SUMMARY

getAACandidate(X500Principal) Returns an array of issuer candidates, by performing a name comparison of the AC's issuer and the subject names of the certificates in the trust

store.

16.14.5 METHODS

- **getAACandidate**
  `java.security.cert.X509Certificate[]` **getAACandidate**`( javax.security.auth.x500.X500Principa`
  issuer `)`

  – **Description**

  Returns an array of issuer candidates, by performing a name comparison of the AC's issuer and the subject names of the certificates in the trust store.
  NOTE: No actual verification or validation of signature takes place in this function.

  – **Parameters**

  * `issuer` – the principal to find an issuer for. If `null`, all known AAs will be returned.

  – **Returns** – an array of issuer candidates, or `null` in case of an error.

16.15 INTERFACE VOMSTRUSTSTORE

16.15.1 DECLARATION

public interface VOMSTrustStore

16.15.2 ALL KNOWN SUBINTERFACES

PKIStore (in 16.5, page 94)

16.15.3 ALL CLASSES KNOWN TO IMPLEMENT INTERFACE

PKIStore (in 16.5, page 94)

16.15.4 METHOD SUMMARY

getAACandidate(X500Principal, String) Returns candidates to the role of signer of an AC with he given issuer and of the give VO.
getLSC(String, String) Returns the LSCFile corresponding to the VO and Host specified.
stopRefresh() Stops refreshing the store.

16.15.5 METHODS

- **getAACandidate**
  `java.security.cert.X509Certificate[]` **getAACandidate**`( javax.security.auth.x500.X500Principa`
  issuer`, java.lang.String` **voName** `)`

  – **Description**

  Returns candidates to the role of signer of an AC with he given issuer and of the give VO.

  - **Parameters**
    - `issuer` – the DN of the signer.
    - `voName` – the VO to which he signer belongs.
  - **Returns** – an array of issuer candidates, or null if none is found.

- **getLSC**
  `org.glite.voms.LSCFile` **getLSC**`( java.lang.String` **voName**`, java.lang.String` **hostName** `)`
  - **Description**
    Returns the LSCFile corresponding to the VO and Host specified.
  - **Parameters**
    - `voName` – the name of the VO.
    - `hostName` – the name of the issuing host.
  - **Returns** – the LSCfile, or null if none is found.

- **stopRefresh**
  `void` **stopRefresh**`( )`
  - **Description**
    Stops refreshing the store. This method MUST be called prior to disposing of the store.

16.16 CLASS ACCERTS

This class represents the ACCerts extension which may be present in the AC.

16.16.1 DECLARATION

public class ACCerts
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.16.2 CONSTRUCTOR SUMMARY

ACCerts() Creates an empty ACCerts object.
ACCerts(ASN1Sequence) Creates an ACCerts starting from a sequence.

16.16.3 METHOD SUMMARY

addCert(X509CertificateStructure) Manually adds a certificate to the list.
getCerts() Gets the certificates.
getDERObject() Makes a DERObject representation.
getInstance(ASN1Sequence) static variant of the constructor.

16.16.4 CONSTRUCTORS

- **ACCerts**
  `public` **ACCerts**`( )`
  - **Description**
    Creates an empty ACCerts object.
- **ACCerts**
  `public` **ACCerts**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Creates an ACCerts starting from a sequence.
  - **Parameters**
    * `seq` – the Sequence.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if Certificates are not supported or if there is an encoding error.

16.16.5 METHODS

- **addCert**
  `public void` **addCert**`( org.bouncycastle.asn1.x509.X509CertificateStructure` **cert** `)`
  - **Description**
    Manually adds a certificate to the list.
  - **Parameters**
    * `cert` – The certificate to add.
- **getCerts**
  `public java.util.List` **getCerts**`( )`
  - **Description**
    Gets the certificates.
  - **Returns** – the list of certificates.
- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Makes a DERObject representation.
  - **Returns** – the DERObject
- **getInstance**
  `public static ACCerts` **getInstance**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    static variant of the constructor.
  - **See also**
    * `ACCerts(org.bouncycastle.asn1.ASN1Sequence)` (in 16.16.4, page 128)

16.17 CLASS ACGENERATOR

```
AttributeCertificateInfo ::= SEQUENCE {
     version              AttCertVersion -- version is v2,
     holder               Holder,
     issuer               AttCertIssuer,
     signature            AlgorithmIdentifier,
     serialNumber         CertificateSerialNumber,
     attrCertValidityPeriod   AttCertValidityPeriod,
     attributes           SEQUENCE OF Attribute,
     issuerUniqueID       UniqueIdentifier OPTIONAL,
     extensions           Extensions OPTIONAL
}

AttCertVersion ::= INTEGER { v2(1) }
```

16.17.1 DECLARATION

public class ACGenerator
extends java.lang.Object

16.17.2 CONSTRUCTOR SUMMARY

ACGenerator()

16.17.3 METHOD SUMMARY

addAttribute(String, String, String)
addAttributes(String, String, List)
generateACInfo()
setExtensions(Vector)
setHolderIssuer(X500Principal)
setHolderSerial(BigInteger)
setIssuer(X500Principal)
setNotAfter(Date)
setNotBefore(Date)
sign(PrivateKey)

16.17.4 CONSTRUCTORS

- **ACGenerator**
  `public` **ACGenerator**`( )`

16.17.5 METHODS

- **addAttribute**
  `public void` **addAttribute**`( java.lang.String` **oid**`, java.lang.String` **policyAuthority**`, java.lang.String` **value** `)`
  - **Parameters**
    * `oid` –
    * `policyAuthority` –
    * `value` –
- **addAttributes**
  `public void` **addAttributes**`( java.lang.String` **oid**`, java.lang.String` **policyAuthority**`, java.util.List` **values** `)`
  - **Parameters**
    * `oid` –
    * `policyAuthority` –
    * `values` –
- **generateACInfo**
  `public AttributeCertificateInfo` **generateACInfo**`( )`
- **setExtensions**
  `public void` **setExtensions**`( java.util.Vector` **vector** `)`
  - **Parameters**
    * `vector` –
- **setHolderIssuer**
  `public void` **setHolderIssuer**`( javax.security.auth.x500.X500Principal` **principal** `)`
  - **Parameters**
    * `principal` –
- **setHolderSerial**
  `public void` **setHolderSerial**`( java.math.BigInteger` **integer** `)`
  - **Parameters**
    * `integer` –
- **setIssuer**
  `public void` **setIssuer**`( javax.security.auth.x500.X500Principal` **principal** `)`
  - **Parameters**
    * `principal` –
- **setNotAfter**
  `public void` **setNotAfter**`( java.util.Date` **date** `)`
  - **Parameters**

* date –

- **setNotBefore**
  `public void` **setNotBefore**`( java.util.Date` **date** `)`

  – **Parameters**

  * date –

- **sign**
  `public void` **sign**`( java.security.PrivateKey` **key** `)`

16.18 CLASS ACTARGET

The intent of this class is to represent a single target.

16.18.1 DECLARATION

public class ACTarget
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.18.2 CONSTRUCTOR SUMMARY

ACTarget() empty contructor
ACTarget(ASN1Sequence) Creates an ACTarget from a sequence

16.18.3 METHOD SUMMARY

getDERObject() Makes a DERObject representation.
getGroup() Gets the group element
getInstance(ASN1Sequence) Static variant of the constructor.
getIssuerSerial() Gets the IssuerSerial
getIssuerSerialString() Gets the IssuerSerial element
getName() Gets the name element.
setGroup(GeneralName) Sets the group.
setGroup(String) Sets the group
setIssuerSerial(IssuerSerial) Sets the IssuerSerial
setIssuerSerial(String) Sets the IssuerSerial
setName(GeneralName) Sets the name
setName(String) Sets the name
toString() Creates a string representation of the target.

16.18.4 CONSTRUCTORS

- **ACTarget**
  `public` **ACTarget**`( )`

  - **Description**
    empty contructor

- **ACTarget**
  `public` **ACTarget**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Creates an ACTarget from a sequence
  - **Parameters**
    * `seq` – the Sequence
  - **Throws**
    * `java.lang.IllegalArgumentException` – if there are parsing problems.

16.18.5 METHODS

- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Makes a DERObject representation.
  - **Returns** – the DERObject
- **getGroup**
  `public java.lang.String` **getGroup**`( )`
  - **Description**
    Gets the group element
  - **Returns** – the group.
- **getInstance**
  `public static ACTarget` **getInstance**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Static variant of the constructor.
  - **See also**
    * `ACTarget(org.bouncycastle.asn1.ASN1Sequence)` (in 16.18.4, page 132)
- **getIssuerSerial**
  `public org.bouncycastle.asn1.x509.IssuerSerial` **getIssuerSerial**`( )`
  - **Description**
    Gets the IssuerSerial
  - **Returns** – the IssuerSerial
- **getIssuerSerialString**
  `public java.lang.String` **getIssuerSerialString**`( )`
  - **Description**
    Gets the IssuerSerial element

  - **Returns** – the IssuerSerial as String.

- **getName**
  `public java.lang.String` **getName**`( )`
  - **Description**
    Gets the name element.
  - **Returns** – the name.

- **setGroup**
  `public void` **setGroup**`( org.bouncycastle.asn1.x509.GeneralName` **g** `)`
  - **Description**
    Sets the group.
  - **Parameters**
    * `g` – the group

- **setGroup**
  `public void` **setGroup**`( java.lang.String` **s** `)`
  - **Description**
    Sets the group
  - **Parameters**
    * `s` – the group name.

- **setIssuerSerial**
  `public void` **setIssuerSerial**`( org.bouncycastle.asn1.x509.IssuerSerial` **is** `)`
  - **Description**
    Sets the IssuerSerial
  - **Parameters**
    * `is` – the IssuerSerial

- **setIssuerSerial**
  `public void` **setIssuerSerial**`( java.lang.String` **s** `)`
  - **Description**
    Sets the IssuerSerial
  - **Parameters**
    * `s` – a textual representation of the IssuerSerial, in the from subject:serial

- **setName**
  `public void` **setName**`( org.bouncycastle.asn1.x509.GeneralName` **n** `)`
  - **Description**
    Sets the name
  - **Parameters**
    * `n` – the name

- **setName**
  `public void` **setName**`( java.lang.String` **s** `)`
  - **Description**
    Sets the name
  - **Parameters**
    * `s` – the name.

- **toString**
  `public java.lang.String` **toString**`( )`
  - **Description**
    Creates a string representation of the target.
  - **Returns** – the string, or null if there were problems.

16.19 CLASS ACTARGETS

The intent of this class is to represent the ACTargets extension which may be present in the AC.

16.19.1 DECLARATION

public class ACTargets
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.19.2 CONSTRUCTOR SUMMARY

ACTargets() Empty constructor.
ACTargets(ASN1Sequence) Creates an ACTargets from a sequence.

16.19.3 METHOD SUMMARY

AddTarget(ACTarget) Manually add a target.
addTarget(String) Manually add a target.
getDERObject() Makes a DERObject representation.
getInstance(ASN1Sequence) Static variant of the constructor.
getTargets() Gets the list of targets.

16.19.4 CONSTRUCTORS

- **ACTargets**
  `public` **ACTargets**`( )`
  - **Description**
    Empty constructor.

- **ACTargets**
  `public` **ACTargets**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Creates an ACTargets from a sequence.
  - **Parameters**
    * `seq` – the sequence.
  - **Throws**
    * `java.lang.IllegalArgumentException` – if there are parsing errors.

16.19.5 METHODS

- **AddTarget**
  `public void` **AddTarget**`( ACTarget` **act** `)`
  - **Description**
    Manually add a target.
  - **Parameters**
    * `act` – the target.
  - **See also**
    * `ACTarget` (in 16.18, page 131)
- **addTarget**
  `public void` **addTarget**`( java.lang.String` **s** `)`
  - **Description**
    Manually add a target.
  - **Parameters**
    * `s` – the target.
- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Makes a DERObject representation.
  - **Returns** – the DERObject
- **getInstance**
  `public static ACTargets` **getInstance**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Static variant of the constructor.
  - **See also**
    * `ACTargets(org.bouncycastle.asn1.ASN1Sequence)` (in 16.19.4, page 135)
- **getTargets**
  `public java.util.List` **getTargets**`( )`

– **Description**
Gets the list of targets.
– **Returns** – a List containing the targets, expressed as String.

16.20 CLASS ACVALIDATOR

Validator class capable of validating an Attribute Certificate and verify its signature against a trust store of Attribute Authority certificates.

16.20.1 DECLARATION

public class ACValidator
extends java.lang.Object

16.20.2 FIELD SUMMARY

log
myTrustStore
myVOMSStore
theVerifier

16.20.3 CONSTRUCTOR SUMMARY

ACValidator(ACTrustStore)
ACValidator(VOMSTrustStore)

16.20.4 METHOD SUMMARY

cleanup()
getInstance()
getInstance(ACTrustStore)
getInstance(VOMSTrustStore)
validate(AttributeCertificate)

16.20.5 FIELDS

- protected static org.apache.log4j.Logger **log**
- protected ACTrustStore **myTrustStore**
- protected VOMSTrustStore **myVOMSStore**
- protected org.glite.voms.PKIVerifier **theVerifier**

16.20.6 CONSTRUCTORS

- **ACValidator**
  `public` **ACValidator**`( ACTrustStore` **trustStore** `)`
- **ACValidator**
  `public` **ACValidator**`( VOMSTrustStore` **theStore** `)`

16.20.7 METHODS

- **cleanup**
  `public void` **cleanup**`( )`
- **getInstance**
  `public static ACValidator` **getInstance**`( )`
- **getInstance**
  `public static ACValidator` **getInstance**`( ACTrustStore` **trustStore** `) throws java.lang.IllegalArg`
- **getInstance**
  `public static ACValidator` **getInstance**`( VOMSTrustStore` **trustStore** `) throws java.lang.IllegalA`
- **validate**
  `public boolean` **validate**`( AttributeCertificate` **ac** `)`

16.21 CLASS ATTCERTISSUER

Shadow implementation of AttributeCertificateInfo from BouncyCastle

16.21.1 DECLARATION

public class AttCertIssuer
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.21.2 CONSTRUCTOR SUMMARY

AttCertIssuer(DEREncodable)
AttCertIssuer(GeneralNames)
AttCertIssuer(V2Form)

16.21.3 METHOD SUMMARY

getDERObject() Produce an object suitable for an ASN1OutputStream.
getIssuerName()

16.21.4 CONSTRUCTORS

- **AttCertIssuer**
  `public` **AttCertIssuer**`( org.bouncycastle.asn1.DEREncodable` **obj** `)`
- **AttCertIssuer**
  `public` **AttCertIssuer**`( org.bouncycastle.asn1.x509.GeneralNames` **v1FormIn** `)`
- **AttCertIssuer**
  `public` **AttCertIssuer**`( V2Form` **v2FormIn** `)`

16.21.5 METHODS

- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Produce an object suitable for an ASN1OutputStream.

```
AttCertIssuer ::= CHOICE {
     v1Form   GeneralNames,  - MUST NOT be used in this
                             - profile
     v2Form   [0] V2Form     - v2 only
}
```

- **getIssuerName**
  `public org.bouncycastle.asn1.x509.GeneralNames` **getIssuerName**`( )`

16.22 CLASS ATTRIBUTECERTIFICATE

A shadow implementation of the non-working BouncyCastle implementation of X.509 Attribute Certificates

16.22.1 DECLARATION

public class AttributeCertificate
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.22.2 FIELD SUMMARY

logger

16.22.3 CONSTRUCTOR SUMMARY

AttributeCertificate(ASN1Sequence)

16.22.4 METHOD SUMMARY

getAcinfo()
getAttributes()
getAttributes(String) Returns a list of the attributes matching the provided OID.
getCertList()
getDERObject() Produce an object suitable for an ASN1OutputStream.
getExtensions()
getFullAttributes()
getFullyQualifiedAttributes()
getHolder()
getHolderX509()
getHost()
getHostPort()
getInstance(InputStream) Create an Attribute Certificate from a input stream containing DER-encoded data
getIssuer()
getIssuerX509()
getListOfFQAN()
getNotAfter()
getNotBefore()
getPort()
getSerialNumber()
getSignature()
getSignatureAlgorithm()
getSignatureValue()
getTargets()
getVO()
isValid() Synonym for `validAt(null)`
validAt(Date) Checks if the AC was valid at the provided timestamp.
verify(PublicKey) Verifies the signature of the AC using the provided signature key
verifyCert(X509Certificate)

16.22.5 FIELDS

- protected static org.apache.log4j.Logger **logger**

16.22.6 CONSTRUCTORS

- **AttributeCertificate**
  `public` **AttributeCertificate**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `) throws java.io.IOExcepti`

16.22.7 METHODS

- **getAcinfo**
  `public AttributeCertificateInfo` **getAcinfo**`( )`

- **getAttributes**
  `public org.bouncycastle.asn1.ASN1Sequence` **getAttributes**`( )`
  - **See also**
    * `AttributeCertificateInfo.getAttributes()` (in 16.23.7, page 144)
- **getAttributes**
  `public java.util.List` **getAttributes**`( java.lang.String` **oid** `)`
  - **Description**
    Returns a list of the attributes matching the provided OID.
  - **Parameters**
    * `oid` – Object Identifier, on the form "1.2.3.4"
  - **Returns** – List of ASN.1 objects representing the OID type in question
- **getCertList**
  `public ACCerts` **getCertList**`( )`
- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Produce an object suitable for an ASN1OutputStream.

```
AttributeCertificate ::= SEQUENCE {
     acinfo               AttributeCertificateInfo,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING
}
```

- **getExtensions**
  `public org.bouncycastle.asn1.x509.X509Extensions` **getExtensions**`( )`
- **getFullAttributes**
  `public FullAttributes` **getFullAttributes**`( )`
- **getFullyQualifiedAttributes**
  `public java.util.List` **getFullyQualifiedAttributes**`( )`
  - **Returns** – List of String of the VOMS fully qualified attributes names (FQANs):
    `vo[/group[/group2...]][/Role=[role]][/Capability=capability]`
- **getHolder**
  `public Holder` **getHolder**`( )`
- **getHolderX509**
  `public java.lang.String` **getHolderX509**`( )`
- **getHost**
  `public java.lang.String` **getHost**`( )`

- **getHostPort**
  `public java.lang.String` **getHostPort**`( )`
- **getInstance**
  `public static AttributeCertificate` **getInstance**`( java.io.InputStream` **in** `) throws java.io.IOException`
  - **Description**
    Create an Attribute Certificate from a input stream containing DER-encoded data
  - **Parameters**
    * `in` –
  - **Returns** – the Attribute Certificate
  - **Throws**
    * `java.io.IOException` –
- **getIssuer**
  `public javax.security.auth.x500.X500Principal` **getIssuer**`( )`
- **getIssuerX509**
  `public org.bouncycastle.jce.X509Principal` **getIssuerX509**`( )`
- **getListOfFQAN**
  `public java.util.List` **getListOfFQAN**`( )`
  - **Returns** – List of FQAN of the VOMS fully qualified attributes names (FQANs)
  - **See also**
    * `org.glite.voms.FQAN` (in 16.2, page 91)
- **getNotAfter**
  `public java.util.Date` **getNotAfter**`( ) throws java.text.ParseException`
- **getNotBefore**
  `public java.util.Date` **getNotBefore**`( ) throws java.text.ParseException`
- **getPort**
  `public int` **getPort**`( )`
- **getSerialNumber**
  `public org.bouncycastle.asn1.DERInteger` **getSerialNumber**`( )`
- **getSignature**
  `public byte[]` **getSignature**`( )`
- **getSignatureAlgorithm**
  `public org.bouncycastle.asn1.x509.AlgorithmIdentifier` **getSignatureAlgorithm**`( )`
- **getSignatureValue**
  `public org.bouncycastle.asn1.DERBitString` **getSignatureValue**`( )`
- **getTargets**
  `public ACTargets` **getTargets**`( )`

- **getVO**
  `public java.lang.String` **getVO**`( )`
- **isValid**
  `public boolean` **isValid**`( )`
  - **Description**
    Synonym for `validAt(null)`
  - **Returns** – true if currently valid
- **validAt**
  `public boolean` **validAt**`( java.util.Date` **date** `)`
  - **Description**
    Checks if the AC was valid at the provided timestamp.
  - **Parameters**
    * `date` – if `null`, current time is used
  - **Returns** – true if the AC was valid at the time in question.
- **verify**
  `public boolean` **verify**`( java.security.PublicKey` **key** `)`
  - **Description**
    Verifies the signature of the AC using the provided signature key
  - **Parameters**
    * `key` – The (RSA) public key to verify the signature with
  - **Returns** – `true` if success, `false` otherwise
- **verifyCert**
  `public boolean` **verifyCert**`( java.security.cert.X509Certificate` **cert** `)`

16.23 CLASS ATTRIBUTECERTIFICATEINFO

Shadow implementation of AttributeCertificateInfo from BouncyCastle

16.23.1 DECLARATION

public class AttributeCertificateInfo
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.23.2 FIELD SUMMARY

AC_CERTS_OID
AC_FULL_ATTRIBUTES_OID
AC_TARGET_OID
VOMS_ATTR_OID
VOMS_EXT_OID

16.23.3 CONSTRUCTOR SUMMARY

AttributeCertificateInfo(ASN1Sequence)

16.23.4 METHOD SUMMARY

getAttCertVersion()
getAttrCertValidityPeriod()
getAttributes()
getCertList()
getDERObject() Produce an object suitable for an ASN1OutputStream.
getExtensions()
getFullAttributes()
getFullyQualifiedAttributes()
getHolder()
getHost()
getHostPort()
getInstance(ASN1Sequence)
getIssuer()
getIssuerUniqueID()
getListOfFQAN()
getPort()
getSerialNumber()
getSignature()
getTargets()
getVO()

16.23.5 FIELDS

- public static final java.lang.String **AC_TARGET_OID**
- public static final java.lang.String **AC_CERTS_OID**
- public static final java.lang.String **AC_FULL_ATTRIBUTES_OID**
- public static final java.lang.String **VOMS_EXT_OID**
- public static final java.lang.String **VOMS_ATTR_OID**

16.23.6 CONSTRUCTORS

- **AttributeCertificateInfo**
  `public` **AttributeCertificateInfo**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`

16.23.7 METHODS

- **getAttCertVersion**
  `public org.bouncycastle.asn1.DERInteger` **getAttCertVersion**`( )`

- **getAttrCertValidityPeriod**
  `public org.bouncycastle.asn1.x509.AttCertValidityPeriod` **getAttrCertValidityPeriod**`( )`
- **getAttributes**
  `public org.bouncycastle.asn1.ASN1Sequence` **getAttributes**`( )`
- **getCertList**
  `public ACCerts` **getCertList**`( )`
- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Produce an object suitable for an ASN1OutputStream.

```
AttributeCertificateInfo ::= SEQUENCE {
     version              AttCertVersion -- version is v2,
     holder               Holder,
     issuer               AttCertIssuer,
     signature            AlgorithmIdentifier,
     serialNumber         CertificateSerialNumber,
     attrCertValidityPeriod   AttCertValidityPeriod,
     attributes           SEQUENCE OF Attribute,
     issuerUniqueID       UniqueIdentifier OPTIONAL,
     extensions           Extensions OPTIONAL
}

AttCertVersion ::= INTEGER { v2(1) }
```

- **getExtensions**
  `public org.bouncycastle.asn1.x509.X509Extensions` **getExtensions**`( )`
- **getFullAttributes**
  `public FullAttributes` **getFullAttributes**`( )`
- **getFullyQualifiedAttributes**
  `public java.util.List` **getFullyQualifiedAttributes**`( )`
  - **Returns** – List of String of the VOMS fully qualified attributes names (FQANs):
    `vo[/group[/group2...]][/Role=[role]][/Capability=capability]`
- **getHolder**
  `public Holder` **getHolder**`( )`

- **getHost**
  `public java.lang.String` **getHost**`( )`
- **getHostPort**
  `public java.lang.String` **getHostPort**`( )`
- **getInstance**
  `public static AttributeCertificateInfo` **getInstance**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
- **getIssuer**
  `public AttCertIssuer` **getIssuer**`( )`
- **getIssuerUniqueID**
  `public org.bouncycastle.asn1.DERBitString` **getIssuerUniqueID**`( )`
- **getListOfFQAN**
  `public java.util.List` **getListOfFQAN**`( )`
  - **Returns** – List of FQAN of the VOMS fully qualified attributes names (FQANs)
  - **See also**
    * `org.glite.voms.FQAN` (in 16.2, page 91)
- **getPort**
  `public int` **getPort**`( )`
- **getSerialNumber**
  `public org.bouncycastle.asn1.DERInteger` **getSerialNumber**`( )`
- **getSignature**
  `public org.bouncycastle.asn1.x509.AlgorithmIdentifier` **getSignature**`( )`
- **getTargets**
  `public ACTargets` **getTargets**`( )`
- **getVO**
  `public java.lang.String` **getVO**`( )`

16.24 CLASS ATTRIBUTEHOLDER

This calss represents an Attribute Holder object.

16.24.1 DECLARATION

public class AttributeHolder
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.24.2 CONSTRUCTOR SUMMARY

AttributeHolder() Empty constructor.
AttributeHolder(ASN1Sequence) Creates an AttributeHolder object from a Sequence.

16.24.3 METHOD SUMMARY

getAttributes() Gets a list of Generic Attributes.
getDERObject() Makes a DERObject representation.
getGrantor() Gets the Grantor of these attributes.
getInstance(ASN1Sequence) Static variant of the constructor.

16.24.4 CONSTRUCTORS

- **AttributeHolder**
  `public` **AttributeHolder**`( )`
  - **Description**
    Empty constructor.
- **AttributeHolder**
  `public` **AttributeHolder**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Creates an AttributeHolder object from a Sequence.
  - **Parameters**
    * `seq` – the Sequence
  - **Throws**
    * `java.lang.IllegalArgumentException` – if there are parsing problems.

16.24.5 METHODS

- **getAttributes**
  `public java.util.List` **getAttributes**`( )`
  - **Description**
    Gets a list of Generic Attributes.
  - **Returns** – the list or null if none was loaded.
- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Makes a DERObject representation.
  - **Returns** – the DERObject
- **getGrantor**
  `public java.lang.String` **getGrantor**`( )`

- **Description**
  Gets the Grantor of these attributes.
- **Returns** – the grantor.

- **getInstance**
  `public static AttributeHolder` **getInstance**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Static variant of the constructor.
  - **See also**
    * `AttributeHolder(org.bouncycastle.asn1.ASN1Sequence)` (in 16.24.4, page 146)

16.25 CLASS FULLATTRIBUTES

This class represents the GenericAttributes extension which may be found in the AC.

16.25.1 DECLARATION

public class FullAttributes
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.25.2 CONSTRUCTOR SUMMARY

FullAttributes() Empty contructor
FullAttributes(ASN1Sequence) Creates a FullAttributes object from a sequence.

16.25.3 METHOD SUMMARY

getAttributeHolders() Returns a list of the AttributeHolders.
getDERObject() Makes a DERObject representation.
getInstance(ASN1Sequence) Static variant of the constructor.

16.25.4 CONSTRUCTORS

- **FullAttributes**
  `public` **FullAttributes**`( )`
  - **Description**
    Empty contructor

- **FullAttributes**
  `public` **FullAttributes**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Creates a FullAttributes object from a sequence.

- **Parameters**
  - `seq` – the Sequence
- **Throws**
  - `java.lang.IllegalArgumentException` – if there are parsing problems.

16.25.5 METHODS

- **getAttributeHolders**
  `public java.util.List` **getAttributeHolders**`( )`
  - **Description**
    Returns a list of the AttributeHolders.
  - **Returns** – the list or null if none was there.
- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Makes a DERObject representation.
  - **Returns** – the DERObject
- **getInstance**
  `public static FullAttributes` **getInstance**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Static variant of the constructor.
  - **See also**
    - `FullAttributes(org.bouncycastle.asn1.ASN1Sequence)` (in 16.25.4, page 147)

16.26 CLASS GENERICATTRIBUTE

This class represents the single Generic Attribute.

16.26.1 DECLARATION

public class GenericAttribute
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.26.2 CONSTRUCTOR SUMMARY

GenericAttribute() Empty contructor
GenericAttribute(ASN1Sequence) Creates a GenericAttributes object from a sequence.

16.26.3 METHOD SUMMARY

getDERObject() Makes a DERObject representation.
getInstance(ASN1Sequence) Static variant of the constructor.
getName() Gets the name of the attribute
getQualifier() Gets the qualifier of the attribute
getValue() Gets the value of the attribute

16.26.4 CONSTRUCTORS

- **GenericAttribute**
  `public` **GenericAttribute**`( )`
  - **Description**
    Empty contructor
- **GenericAttribute**
  `public` **GenericAttribute**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Creates a GenericAttributes object from a sequence.
  - **Parameters**
    * `seq` – the Sequence
  - **Throws**
    * `java.lang.IllegalArgumentException` – if there are parsing problems.

16.26.5 METHODS

- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Makes a DERObject representation.
  - **Returns** – the DERObject
- **getInstance**
  `public static GenericAttribute` **getInstance**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
  - **Description**
    Static variant of the constructor.
  - **See also**
    * `GenericAttribute(org.bouncycastle.asn1.ASN1Sequence)` (in 16.26.4, page 149)
- **getName**
  `public java.lang.String` **getName**`( )`
  - **Description**
    Gets the name of the attribute

  - **Returns** – the name

- **getQualifier**
  `public java.lang.String` **getQualifier**`( )`

  - **Description**
    Gets the qualifier of the attribute
  - **Returns** – the qualifier

- **getValue**
  `public java.lang.String` **getValue**`( )`

  - **Description**
    Gets the value of the attribute
  - **Returns** – the value

16.27 CLASS HOLDER

The Holder object.

```
Holder ::= SEQUENCE {
      baseCertificateID   [0] IssuerSerial OPTIONAL,
               - the issuer and serial number of
               - the holder's Public Key Certificate
      entityName          [1] GeneralNames OPTIONAL,
               - the name of the claimant or role
      objectDigestInfo    [2] ObjectDigestInfo OPTIONAL
               - used to directly authenticate the holder,
               - for example, an executable
}
```

16.27.1 DECLARATION

public class Holder
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.27.2 CONSTRUCTOR SUMMARY

Holder(ASN1Sequence)
Holder(X500Principal, BigInteger)
Holder(X509Certificate)

16.27.3 METHOD SUMMARY

getDERObject()
getIssuer()

isHolder(X509Certificate)
matchesDN(X500Principal, GeneralNames)

16.27.4 CONSTRUCTORS

- **Holder**
  `public` **Holder**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
- **Holder**
  `public` **Holder**`( javax.security.auth.x500.X500Principal` **issuer**`, java.math.BigInteger` **serial** `)`
- **Holder**
  `public` **Holder**`( java.security.cert.X509Certificate` **cert** `)`

16.27.5 METHODS

- **getDERObject**
  `org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
- **getIssuer**
  `public org.bouncycastle.asn1.x509.GeneralNames` **getIssuer**`( )`
- **isHolder**
  `public boolean` **isHolder**`( java.security.cert.X509Certificate` **cert** `)`
- **matchesDN**
  `protected static boolean` **matchesDN**`( javax.security.auth.x500.X500Principal` **subject**`, org.bouncycastle.asn1.x509.GeneralNames` **targets** `)`

16.28 CLASS IETFATTRSYNTAX

Implementation of `IetfAttrSyntax` as specified by RFC3281.

```
IetfAttrSyntax ::= SEQUENCE {
  policyAuthority [0] GeneralNames OPTIONAL,
  values SEQUENCE OF CHOICE {
    octets OCTET STRING,
    oid OBJECT IDENTIFIER,
    string UTF8String
  }
}
```

16.28.1 DECLARATION

public class IetfAttrSyntax
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.28.2 FIELD SUMMARY

VALUE_OCTETS
VALUE_OID
VALUE_UTF8

16.28.3 CONSTRUCTOR SUMMARY

IetfAttrSyntax(ASN1Sequence)

16.28.4 METHOD SUMMARY

getDERObject()
getPolicyAuthority()
getValues()
getValueType()

16.28.5 FIELDS

- public static final int **VALUE_OCTETS**
- public static final int **VALUE_OID**
- public static final int **VALUE_UTF8**

16.28.6 CONSTRUCTORS

- **IetfAttrSyntax**
  `public` **IetfAttrSyntax**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`

16.28.7 METHODS

- **getDERObject**
  `org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
- **getPolicyAuthority**
  `public org.bouncycastle.asn1.x509.GeneralNames` **getPolicyAuthority**`( )`
- **getValues**
  `public java.util.List` **getValues**`( )`
- **getValueType**
  `public int` **getValueType**`( )`

16.29 CLASS OBJECTDIGESTINFO

16.29.1 DECLARATION

public class ObjectDigestInfo
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.29.2 CONSTRUCTOR SUMMARY

ObjectDigestInfo(ASN1Sequence)

16.29.3 METHOD SUMMARY

getDERObject() Produce an object suitable for an ASN1OutputStream.
getDigestAlgorithm()
getDigestedObjectType()
getObjectDigest()
getOtherObjectTypeID()

16.29.4 CONSTRUCTORS

- **ObjectDigestInfo**
  `public` **ObjectDigestInfo**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`

16.29.5 METHODS

- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Produce an object suitable for an ASN1OutputStream.

```
ObjectDigestInfo ::= SEQUENCE {
     digestedObjectType  ENUMERATED {
             publicKey            (0),
             publicKeyCert        (1),
             otherObjectTypes     (2) },
                     - otherObjectTypes MUST NOT
                     - be used in this profile
     otherObjectTypeID   OBJECT IDENTIFIER OPTIONAL,
     digestAlgorithm     AlgorithmIdentifier,
     objectDigest        BIT STRING
}
```

- **getDigestAlgorithm**
  `public org.bouncycastle.asn1.x509.AlgorithmIdentifier` **getDigestAlgorithm**`( )`

- **getDigestedObjectType**
  `public org.bouncycastle.asn1.DEREnumerated` **getDigestedObjectType**`( )`
- **getObjectDigest**
  `public org.bouncycastle.asn1.DERBitString` **getObjectDigest**`( )`
- **getOtherObjectTypeID**
  `public org.bouncycastle.asn1.DERObjectIdentifier` **getOtherObjectTypeID**`( )`

16.30 CLASS UTIL

16.30.1 DECLARATION

public class Util
extends java.lang.Object

16.30.2 CONSTRUCTOR SUMMARY

Util()

16.30.3 METHOD SUMMARY

generalNameToGeneralNames(GeneralName)
generalNameToX500Name(GeneralName)
generalNameToX509Name(GeneralName)
x500nameToGeneralName(byte[])
x500nameToGeneralNames(X500Principal)

16.30.4 CONSTRUCTORS

- **Util**
  `public` **Util**`( )`

16.30.5 METHODS

- **generalNameToGeneralNames**
  `public static org.bouncycastle.asn1.x509.GeneralNames` **generalNameToGeneralNames**`(`
  `org.bouncycastle.asn1.x509.GeneralName` **name** `)`
- **generalNameToX500Name**
  `public static javax.security.auth.x500.X500Principal` **generalNameToX500Name**`( org.bouncycas`
  name `)`
- **generalNameToX509Name**
  `public static org.bouncycastle.jce.X509Principal` **generalNameToX509Name**`( org.bouncycastle`
  name `)`

- **x500nameToGeneralName**
  `public static org.bouncycastle.asn1.x509.GeneralName` **x500nameToGeneralName**`( byte[]` **encodedName** `)`
- **x500nameToGeneralNames**
  `public static org.bouncycastle.asn1.x509.GeneralNames` **x500nameToGeneralNames**`(` `javax.security.auth.x500.X500Principal` **name** `)`

16.31 CLASS V2FORM

16.31.1 DECLARATION

public class V2Form
extends java.lang.Object
implements org.bouncycastle.asn1.DEREncodable

16.31.2 CONSTRUCTOR SUMMARY

V2Form(ASN1Sequence)
V2Form(GeneralNames)

16.31.3 METHOD SUMMARY

getBaseCertificateID()
getDERObject() Produce an object suitable for an ASN1OutputStream.
getIssuerName()
getObjectDigestInfo()

16.31.4 CONSTRUCTORS

- **V2Form**
  `public` **V2Form**`( org.bouncycastle.asn1.ASN1Sequence` **seq** `)`
- **V2Form**
  `public` **V2Form**`( org.bouncycastle.asn1.x509.GeneralNames` **issuerName** `)`

16.31.5 METHODS

- **getBaseCertificateID**
  `public org.bouncycastle.asn1.x509.IssuerSerial` **getBaseCertificateID**`( )`
- **getDERObject**
  `public org.bouncycastle.asn1.DERObject` **getDERObject**`( )`
  - **Description**
    Produce an object suitable for an ASN1OutputStream.
    ```
    V2Form ::= SEQUENCE {
    ```

```
        issuerName            GeneralNames  OPTIONAL,
        baseCertificateID     [0] IssuerSerial  OPTIONAL,
        objectDigestInfo      [1] ObjectDigestInfo  OPTIONAL
          - issuerName MUST be present in this profile
          - baseCertificateID and objectDigestInfo MUST NOT
          - be present in this profile
    }
```

- **getIssuerName**
  `public org.bouncycastle.asn1.x509.GeneralNames` **getIssuerName**`( )`
- **getObjectDigestInfo**
  `public ObjectDigestInfo` **getObjectDigestInfo**`( )`

Package org.glite.voms.contact.cli

Package Contents *Page*

Classes

16.32 CLASS VOMSPROXYINITCLIENT

This class implements a command-line voms-proxy-init client.

16.32.1 DECLARATION

public class VomsProxyInitClient
extends java.lang.Object

16.32.2 CONSTRUCTORS

- *VomsProxyInitClient*
  `public` **VomsProxyInitClient**`( java.lang.String [] args )`

16.32.3 METHODS

- *main*
  `public static void` **main**`( java.lang.String [] args )`

Package org.glite.voms.contact

Package Contents *Page*

Classes

16.33 CLASS MYPROXYCERTINFO

This class represents the ProxyCertInfo extension

16.33.1 DECLARATION

public class MyProxyCertInfo
implements org.bouncycastle.asn1.DEREncodable

16.33.2 CONSTRUCTOR SUMMARY

MyProxyCertInfo(ProxyPolicy, int) Creates an object.
MyProxyCertInfo(int, ProxyPolicy, int) Creates an object.
MyProxyCertInfo(ASN1Sequence) Creates an object.
MyProxyCertInfo(byte[]) Creates an object.

16.33.3 METHOD SUMMARY

getPathLengthConstraint() Returns the path length constraint.
getProxyPolicy() Retruns the proxy policy.

16.33.4 CONSTRUCTORS

- **MyProxyCertInfo**
  `public` **MyProxyCertInfo**`(ProxyPolicy` **policy**`, int` **version**`)`
  - **Description** Initializes a MyProxyCertInfo object
  - **Parameters**
    - **policy** The policy applying to this MyProxyCertInfo
    - **version** The version of the object created. Either GT3 or RFC
- **MyProxyCertInfo**
  `public` **MyProxyCertInfo**`(byte`

  seq`)`
  - **Description** Initializes a MyProxyCertInfo object
  - **Parameters**
    - **seq** The DER representatin of a MyProxyCertInfo
- **MyProxyCertInfo**
  `public` **MyProxyCertInfo**`(int` **pathlen**`, ProxyPolicy` **policy**`, int` **version**`)`
  - **Description** Initializes a MyProxyCertInfo object
  - **Parameters**
    - **pathlen** The value of the PathLenConstraint
    - **policy** The policy applying to this MyProxyCertInfo
    - **version** The version of the object created. Either GT3 or RFC
- **MyProxyCertInfo**
  `public` **MyProxyCertInfo**`(ASN1Sequence` **seq**`)`

- **Description** Initializes a MyProxyCertInfo object
- **Parameters**
  - **seq** The ASN1Sequence to be read ad a MyProxyCertInfo

16.33.5 METHODS

- **getPathLengthConstraint**
  `int` **getPathLengthConstraint**`()`
  - **Description**
    Returns the pathLengthConstraint of the extension.
  - **Returns** – The value of the pathLenghtConstraint.
- **getProxyPolicy**
  `ProxyPolicy` **getProxyPolicy**`()`
  - **Description**
    Returns the proxyPolicy of the extension.
  - **Returns** – The value of the proxyPolicy.

16.34 CLASS PROXYPOLICY

This class represents the ProxyPolicy extension

16.34.1 DECLARATION

public class ProxyPolicy
implements org.bouncycastle.asn1.DEREncodable

16.34.2 CONSTRUCTOR SUMMARY

ProxyPolicy(DERObjectIdentifier) Creates policy.
ProxyPolicy(DERObjectIdentifier, String) Creates policy.
ProxyPolicy(String, String) Creates policy.
ProxyPolicy(String) Creates policy.
ProxyPolicy(ASN1Sequence) Creates policy.

16.34.3 CONSTRUCTORS

- **ProxyPolicy**
  `public` **ProxyPolicy**`(DERObjectIdentifier` **oid**`)`
  - **Description**
    Creates ProxyPolicy with the specified OID
  - **Parameters**
    - `oid` The OID of the policy

- **ProxyPolicy**
  `public` **ProxyPolicy**`(DERObjectIdentifier` **oid**`, String` **policy**`)`
  - **Description**
    Creates ProxyPolicy with the specified OID and policy
  - **Parameters**
    * `oid` The OID of the policy
    * `policy` The text of the policy
- **ProxyPolicy**
  `public` **ProxyPolicy**`(String` **oid**`, String` **policy**`)`
  - **Description**
    Creates ProxyPolicy with the specified OID and policy
  - **Parameters**
    * `oid` The OID of the policy
    * `policy` The text of the policy
- **ProxyPolicy**
  `public` **ProxyPolicy**`(String` **oid**`)`
  - **Description**
    Creates ProxyPolicy with the specified OID
  - **Parameters**
    * `oid` The OID of the policy
- **ProxyPolicy**
  `public` **ProxyPolicy**`(ASN1Sequence` **seq**`)`
  - **Description**
    Creates ProxyPolicy from an ASN.1 Sequence
  - **Parameters**
    * `seq` The ASN.1 sequence

16.35 CLASS VOMSPROXYCONSTANTS

This class defines some constants useful for creatin proxies.

16.35.1 DECLARATION

public class VOMSProxyConstants
extends java.lang.Object

16.35.2 FIELD SUMMARY

DELEGATION_FULL Full delegation.
DELEGATION_NONE No delegation.
DELEGATION_LIMITED Limited Delegation.
GSI_2_LIMITED_PROXY Limited proxy.
GSI_3_LIMITED_PROXY Limited proxy.
GSI_2_PROXY GSI 2 proxy.
GSI_3_IMPERSONATION_PROXY GSI 3 and 4 Impersonation proxy.
GSI_3_RESTRICTED_PROXY GSI 3 and 4 Restricted proxy.
GSI_3_INDEPENDENT_PROXY GSI 3 and 4 Independent proxy.

16.35.3 FIELDS

- public static final int **DELEGATION_FULL**
  - Proxy allows Full delegation
- public static final int **DELEGATION_NONE**
  - Proxy does not allow delegation
- public static final int **DELEGATION_LIMITED**
  - Proxy allows limited delegation
- public static final int **GSI_2_LIMITED_PROXY**
  - GT2 Limited proxy
- public static final int **GSI_3_LIMITED_PROXY**
  - GT3 Limited proxy
- public static final int **GSI_2_PROXY**
  - GT2 proxy
- public static final int **GSI_3_IMPERSONATION_PROXY**
  - GT 3 or 4 impersonation proxy
- public static final int **GSI_3_RESTRICTED_PROXY**
  - GT 3 or 4 restricted proxy
- public static final int **GSI_3_INDEPENDENT_PROXY**
  - GT 3 or 4 independent proxy

16.36 CLASS PATHNAMINGSCHEME

This class provides utility methods that are used for parsing, matching voms FQANs (Fully Qualified Attribute Names).

16.36.1 DECLARATION

public class PathNamingScheme

extends java.lang.Object

16.36.2 FIELDS

- public static final Logger log
  –
- public static final String containerSyntax
  –
- public static final String groupSyntax
  –
- public static final String roleSyntax
  –
- public static final String qualifiedRoleSyntax
  –
- public static final String capabilitySyntax
  –
- public static final Pattern containerPattern
  –
- public static final Pattern groupPattern
  –
- public static final Pattern rolePattern
  –
- public static final Pattern qualifiedRolePattern
  –
- public static final Pattern capabilityPattern
  –

16.36.3 CONSTRUCTORS

- *PathNamingScheme*
  `public` **PathNamingScheme**`( )`

16.36.4 METHODS

- *checkGroup*
  `public static void` **checkGroup**`( java.lang.String` **groupName** `)`
  - **Usage**
    - This methods checks that the string passed as argument complies with the syntax used by voms to identify groups.
  - **Parameters**
    - `groupName,` - the string that has to be checked.
  - **Exceptions**
    - `org.glite.voms.contact.VOMSSyntaxException` - If the string passed as argument doens not comply with the voms sytax.

- *checkRole*
  `public static void` **checkRole**`( java.lang.String` **roleName** `)`
  - **Usage**
    - This methods checks that the string passed as argument complies with the syntax used by voms to identify roles.
  - **Parameters**
    - `roleName` -
  - **Exceptions**
    - `org.glite.voms.contact.VOMSSyntaxException` - If the string passed as argument doens not comply with the voms sytax.

- *checkSyntax*
  `public static void` **checkSyntax**`( java.lang.String` **containerName** `)`
  - **Usage**
    - This methods checks that the string passed as argument complies with the voms FQAN syntax.
  - **Parameters**
    - `containerName,` - the string that must be checked for compatibility with FQAN syntax.
  - **Exceptions**
    - `org.glite.voms.contact.VOMSSyntaxException` - If there's an error in the FQAN syntax.

- *getGroupName*
  `public static String` **getGroupName**`( java.lang.String` **containerName** `)`
  - **Usage**
    - This method extracts group name information from the FQAN passed as argument.
  - **Parameters**
    - `containerName,` - the FQAN
  - **Returns** -
    - A string containing the group name, if found
    - null, if no group information is contained in the FQAN passed as argument

- *getRoleName*
  `public static String` **getRoleName**`( java.lang.String` **containerName** `)`
  - **Usage**
    - This method extracts the role name information from the FQAN passed as argument.
  - **Parameters**
    - `containerName,` - the FQAN
  - **Returns** -
    - A string containing the role name, if found
    - null, if no role information is contained in the FQAN passed as argument

- *isGroup*
  `public static boolean` **isGroup**`( java.lang.String` **groupName** `)`
  - **Usage**
    - This methods checks that the FQAN passed as argument identifies a voms group.
  - **Parameters**
    - `groupName,` - the string to check.
  - **Returns** -
    - true, if the string passed as argument identifies a voms group.
    - false, otherwise.

- *isQualifiedRole*
  `public static boolean` **isQualifiedRole**`( java.lang.String` **roleName** `)`
  - **Usage**
    - This methods checks that the FQAN passed as argument identifies a qualified voms role, i.e., a role defined in the context of a voms group.
  - **Parameters**
    - `roleName,` - the string to check.
  - **Returns** -
    - true, if the string passed as argument identifies a qualified voms role.
    - false, otherwise.

- *isRole*
  `public static boolean` **isRole**`( java.lang.String` **roleName** `)`
  - **Usage**
    - This methods checks that the FQAN passed as argument identifies a voms role.
  - **Parameters**
    - `roleName,` - the string to check.
  - **Returns** -
    - true, if the string passed as argument identifies a voms role.
    - false, otherwise.

- *toOldQualifiedRoleSyntax*
  `public static String` **toOldQualifiedRoleSyntax**`( java.lang.String` **qualifiedRole** `)`

16.37 CLASS TEST

16.37.1 DECLARATION

public class Test

extends java.lang.Object

16.37.2 CONSTRUCTORS

- *Test*
  `public` **Test**`( )`

16.37.3 METHODS

- *main*
  `public static void` **main**`( java.lang.String [] )` **args**

16.38 CLASS USERCREDENTIALS

This class implements parsing and handling of X509 user credentials in PEM or PKCS12 format.

16.38.1 DECLARATION

public class UserCredentials

extends java.lang.Object

16.38.2 METHODS

- *getUserCertificate*
  `public X509Certificate` **getUserCertificate**`( )`
  - **Usage**
    - This method returs the user certificate loaded in this UserCredentials .
  - **Returns** - the X509 user certificate.
- *save*
  `void` **save**`(OutputStream os)`
  - **Usage**
    itemSaves a copy of the credential in the passed OutputStream
  - **Parameters**
    - `os` The output stream on which to save the credential
  - **Exceptions**
    - **IOException** When the save operation fails
- *getUserChain*
  `public X509Certificate` **getUserChain**`( )`
  - **Usage**

    * This method returs the user certificate chain loaded in this UserCredentials .
  - **Returns** - the X509 user certificate.

- *getUserKey*
  `public BouncyCastleOpenSSLKey` **getUserKey**`( )`
  - **Usage**
    * This method returs the user credential openssl private key.
  - **Returns** - the user credentials private key.

- *instance*
  `public static UserCredentials` **instance**`( )`
  - **Usage**
    * Static instance constructor for a UserCredentials . This method should be used with credentials whose private key is not encrypted.
      The current implementation looks for user credentials in the following places (in sequence):
      · If the `X509_USER_CERT` and `X509_USER_KEY` system properties are set, their values are used to load the user credentials
      · If the `PKCS12_USER_CERT` system property is set, its value is used to load the user credentials.
      · The content of the `.globus` directory in the user's home is searched for a PEM certificate (in the `usercert.pem` and `userkey.pem` files).
      · The content of the .globus directory in the user's home is searched for a PKC12 certificate (in the `usercert.p12` file).
  - **Returns** - the loaded user credentials.
  - **Exceptions**
    * `org.glite.voms.contact.VOMSException` - if there is an error loading the user credentials.

- *instance*
  `public static UserCredentials` **instance**`( java.lang.String` **keyPassword** `)`
  - **Usage**
    * Static instance constructor for a UserCredentials . For more info on the user credentials load procedure, see #instance() .
  - **Parameters**
    * `keyPassword`, - the password that is to be used to decrypt the user private key.
  - **Returns** - the loaded user credentials.
  - **Exceptions**
    * `org.glite.voms.contact.VOMSException` - if there is an error loading the user credentials.

- *instance*
  `public static UserCredentials` **instance**`( java.lang.String` **userCertFile**`, java.lang.String` **userKeyFile** `)`
  - **Usage**

* Static instance constructor for a UserCredentials .

  This methods allows a user to bypass the default credentials search procedure (highlighted here), by specifying the path to a PEM X509 user cert and private key.

– **Parameters**
  * `userCertFile`, - the path to the PEM X509 user certificate.
  * `userKeyFile`, - the path to the PEM X509 private key.
– **Returns** - the loaded user credentials.
– **Exceptions**
  * `org.glite.voms.contact.VOMSException` - if there is an error loading the user credentials.

- *instance*
  `public static UserCredentials` **instance**`( java.lang.String` **userCertFile**`, java.lang.String` **userKeyFile**`, java.lang.String` **keyPassword** `)`

  – **Usage**
    * Static instance constructor for a UserCredentials .

      This methods allows a user to bypass the default credentials search procedure (highlighted here), by specifying the path to a PEM X509 user cert and private key.
  – **Parameters**
    * `userCertFile`, - the path to the PEM X509 user certificate.
    * `userKeyFile`, - the path to the PEM X509 private key.
    * `keyPassword`, - the password that is to be used to decrypt the user private key.
  – **Returns** - the loaded user credentials.
  – **Exceptions**
    * `org.glite.voms.contact.VOMSException` - if there is an error loading the user credentials.
- *instance*
  `public static UserCredentials` **instance**`( UserCredentials` **credentials**`)`

  – **Usage**
    * Static instance constructor for a UserCredentials .

      This methods allows a user to create a UserCredentials from another UserCredentials.
  – **Parameters**
    * `credentials`, - the source credentials.
  – **Returns** - the loaded user credentials.
  – **Exceptions**
    * `org.glite.voms.contact.VOMSException` - if there is an error loading the user credentials.
- *instance*
  `public static UserCredentials` **instance**`( PrivateKey` **key**`, X509Certificate[]` **cert**`)`

  – **Usage**

* Static instance constructor for a UserCredentials .

  This methods allows a user to create a UserCredential object from an existing private key and certificate chain.

– **Parameters**
  * `key` - the private key of the certificate.
  * `certs` - the certificate chain, including the user certificate.
– **Returns** - the loaded user credentials.
– **Exceptions**
  * `org.glite.voms.contact.VOMSException` - if there is an error loading the user credentials.

16.39 CLASS VOMSDECODER

This class implements a decoder for the non-standard Base-64 algorithm used by voms.

16.39.1 DECLARATION

public class VOMSDecoder

extends java.lang.Object

16.39.2 CONSTRUCTORS

- *VOMSDecoder*
  `public` **VOMSDecoder**`( )`

16.39.3 METHODS

- *decode*
  `public static byte` **decode**`( java.lang.String` **s** `)`

16.40 CLASS VOMSERRORMESSAGE

This class is used to decode VOMS error messages contained in a VOMS response.

16.40.1 DECLARATION

public class VOMSErrorMessage

extends java.lang.Object

16.40.2 CONSTRUCTORS

- *VOMSErrorMessage*
  `public` **VOMSErrorMessage**`( int` **code**`, java.lang.String` **message** `)`

16.40.3 METHODS

- *getCode*
  `public int` **getCode**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *setCode*
  `public void` **setCode**`( int` **code** `)`
- *setMessage*
  `public void` **setMessage**`( java.lang.String` **message** `)`
- *toString*
  `public String` **toString**`( )`

16.41 CLASS VOMSESFILEPARSER

This class implements support for vomses configuration files and directories.

The vomses file search procedure is as follows:

- if the `GLITE_LOCATION` system property is set, the `$GLITE_LOCATION/etc/vomses` path is added to the search path.
- if the `VOMSES_LOCATION` system propery is set, its value its interpreted as a colon (:) separated list of paths that are added to the search path.
- if the `${user.home}/.globus/vomses` file or directory is set, it is added to the search path.
- if the `${user.home}/.glite/vomses` file or directory is set, it is added to the search path.

16.41.1 DECLARATION

public class VOMSESFileParser

extends java.lang.Object

16.41.2 METHODS

- *buildServerMap*
  `public VOMSServerMap` **buildServerMap**`( )`
  - **Usage**
    - This method is used to build a VOMSServerMap object starting from vomses configuration files or directories.
  - **Returns** - a VOMSServerMap object that reflects vomses configuration files.
  - **Exceptions**
    - `java.io.IOException` - if a parsing error occurs, or no vomses file is found.
- *instance*
  `public static VOMSESFileParser` **instance**`( )`
  - **Returns** - a new instance of VOMSESFileParser .

16.42 CLASS VOMSEXCEPTION

16.42.1 DECLARATION

public class VOMSException

extends java.lang.RuntimeException

16.42.2 CONSTRUCTORS

- *VOMSException*
  `public` **VOMSException**`( java.lang.String` **message** `)`
- *VOMSException*
  `public` **VOMSException**`( java.lang.String` **message**`, java.lang.Throwable` **t** `)`
- *VOMSException*
  `public` **VOMSException**`( java.lang.Throwable` **t** `)`

16.42.3 METHODS INHERITED FROM CLASS java.lang.RuntimeException

16.42.4 METHODS INHERITED FROM CLASS java.lang.Exception

16.42.5 METHODS INHERITED FROM CLASS java.lang.Throwable

- *fillInStackTrace*
  `public synchronized native Throwable` **fillInStackTrace**`( )`
- *getCause*
  `public Throwable` **getCause**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *getStackTrace*
  `public StackTraceElement` **getStackTrace**`( )`
- *initCause*
  `public synchronized Throwable` **initCause**`( java.lang.Throwable` **arg0** `)`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream` **arg0** `)`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter` **arg0** `)`
- *setStackTrace*
  `public void` **setStackTrace**`( java.lang.StackTraceElement [] arg0 )`
- *toString*
  `public String` **toString**`( )`

16.43 CLASS VOMSPARSER

This class implements the XML parsing of responses produced by VOMS servers.

16.43.1 DECLARATION

public class VOMSParser

extends java.lang.Object

16.43.2 METHODS

- *instance*
  `public static VOMSParser` **instance**`( )`
  - **Returns** - a new VOMSParser instance.
- *parseResponse*
  `public VOMSResponse` **parseResponse**`( java.io.InputStream` **is** `)`
  - **Usage**
    - Parses a voms response reading from a given input stream.
  - **Parameters**
    - `is,` - the input stream.
  - **Returns** - a VOMSResponse object that represents the parsed response.

16.44 CLASS VOMSPROTOCOL

This class manages the client-side communication protocol with the VOMS server.

16.44.1 DECLARATION

public class VOMSProtocol

extends java.lang.Object

16.44.2 METHODS

- *getResponse*
  `public VOMSResponse` **getResponse**`( java.io.InputStream` **stream** `)`
  - **Usage**
    - This method is used to parse a VOMS response from an input stream.
  - **Parameters**
    - `stream,` - the input stream from which the response will be parsed.
  - **Returns** - a VOMSResponse object.
- *instance*
  `public static VOMSProtocol` **instance**`( )`
- *sendRequest*
  `public void` **sendRequest**`( org.glite.voms.contact.VOMSRequestOptions` **requestOptions**`, java.io.OutputStream` **stream** `)`
  - **Usage**

* This method is used to send a request to a VOMS server.

– **Parameters**

* `requestOptions`, - the request options. See VOMSRequestOptions .
* `stream`, - an output stream.

16.45 CLASS VOMSPROXYBUILDER

This class implements VOMS X509 proxy certificates creation.

16.45.1 DECLARATION

public class VOMSProxyBuilder

extends java.lang.Object

16.45.2 FIELDS

- public static final int GT2_PROXY
  –
- public static final int GT3_PROXY
  –
- public static final int GT4_PROXY
  –
- public static final int DEFAULT_PROXY_TYPE
  –
- public static final int DEFAULT_DELEGATION_TYPE
  –
- public static final int DEFAULT_PROXY_LIFETIME
  –

16.45.3 CONSTRUCTORS

- *VOMSProxyBuilder*
  `public` **VOMSProxyBuilder**`( )`

16.45.4 METHODS

- *buildAC*
  `public static AttributeCertificate` **buildAC**`( byte [] )` **acBytes**

  – **Usage**
    * This methods builds an AttributeCertificate (AC) object starting from an array of bytes.
  – **Parameters**

    - `acBytes`, - the byte array containing the attribute certificate.
  - **Returns** - the AttributeCertificate object
  - **Exceptions**
    - `VOMSException`, - in case of parsing errors.

- *buildProxy*
  `public static GlobusCredential` **buildProxy**`( org.glite.voms.contact.UserCredentials` **cred** `)`
  - **Usage**
    - This methods creates a non-voms proxy using the UserCredentials passed as arguments. The proxy is created with a default lifetime of 3600 seconds.
  - **Parameters**
    - `cred`, - the UserCredentials from which the proxy must be created.
  - **Returns** - a GlobusCredential object that represents the proxy.

- *buildProxy*
  `public static GlobusCredential` **buildProxy**`( org.glite.voms.contact.UserCredentials` **cred**`, int` **lifetime**`, int` **proxyType** `)`
  - **Usage**
    - This methods creates a non-voms proxy using the UserCredentials passed as arguments.
  - **Parameters**
    - `cred`, - the UserCredentials from which the proxy must be created.
    - `lifetime`, - the lifetime (in seconds) of the generated proxy.
  - **Returns** - a GlobusCredential object that represents the proxy.

- *buildProxy*
  `public static GlobusCredential` **buildProxy**`( org.glite.voms.contact.UserCredentials` **cred**`, java.util.List` **ACs**`, int` **lifetime** `)`
  - **Usage**
    - This method is used to create a VOMS proxy starting from the UserCredentials passed as arguments and including a list of AttributeCertificate objects that will be included in the proxy.
  - **Parameters**
    - `cred`, - the UserCredentials from which the proxy must be created.
    - `ACs`, - the list of AttributeCertificate objects.
    - `lifetime`, - the lifetime in seconds of the generated proxy.
  - **Returns** - a GlobusCredential object that represents the proxy.
  - **Exceptions**
    - - VOMSException}, if something goes wrong.

- *buildProxy*
  `public static GlobusCredential` **buildProxy**`( org.glite.voms.contact.UserCredentials` **cred**`, java.util.List` **ACs**`, int` **lifetime**`, int` **gtVersion**`, int` **delegType**`, java.lang.String` **policyType** `)`

- **Usage**
  - This method is used to create a VOMS proxy starting from the UserCredentials passed as arguments and including a list of AttributeCertificate objects that will be included in the proxy.
- **Parameters**
  - `cred`, - the UserCredentials from which the proxy must be created.
  - `ACs`, - the list of AttributeCertificate objects.
  - `lifetime`, - the lifetime in seconds of the generated proxy.
  - `version`, - the version of globus to which the proxy conforms
- **Returns** - a GlobusCredential object that represents the proxy.
- **Exceptions**
  - - VOMSException}, if something goes wrong.

- *saveProxy*
  `public static void` **saveProxy**`( org.globus.gsi.GlobusCredential` **cred**`, java.io.OutputStream` **os** `)`
  - **Usage**
    - This method is write a globus proxy to an output stream.
  - **Parameters**
    - `cred` -
    - `os` -

- *saveProxy*
  `public static void` **saveProxy**`( org.globus.gsi.GlobusCredential` **cred**`, java.lang.String` **filename** `)`
  - **Usage**
    - This method saves a globus proxy to a file.
  - **Parameters**
    - `cred` -
    - `filename` -
  - **Exceptions**
    - `java.io.FileNotFoundException` -

16.46 CLASS VOMSPROXYINIT

This class implements the voms-proxy-init functionality.

16.46.1 DECLARATION

public class VOMSProxyInit

extends java.lang.Object

16.46.2 CONSTRUCTORS

- *VOMSProxyInit*
  `public` **VOMSProxyInit**`( java.lang.String` **privateKeyPassword** `)`

16.46.3 METHODS

- *addVomsServer*
  `public void` **addVomsServer**`( org.glite.voms.contact.VOMSServerInfo` **info** `)`
- *getDelegationType*
  `public int` **getDelegationType**`( )`
- *getPolicyType*
  `public String` **getPolicyType**`( )`
- *getProxyLifetime*
  `public int` **getProxyLifetime**`( )`
- *getProxyOutputFile*
  `public String` **getProxyOutputFile**`( )`
- *getProxyType*
  `public int` **getProxyType**`( )`
- *getVomsAC*
  `public synchronized AttributeCertificate` **getVomsAC**`( org.glite.voms.contact.VOMSRequestOpt` **requestOptions** `)`
- *getVomsProxy*
  `public synchronized GlobusCredential` **getVomsProxy**`( )`
- *getVomsProxy*
  `public synchronized GlobusCredential` **getVomsProxy**`( java.util.Collection` **listOfReqOptions** `)`
- *instance*
  `public static VOMSProxyInit` **instance**`( )`
- *instance*
  `public static VOMSProxyInit` **instance**`( org.globus.gsi.GlobusCredential` **credentials** `)`
- *instance*
  `public static VOMSProxyInit` **instance**`( java.lang.String` **privateKeyPassword** `)`
- *setDelegationType*
  `public void` **setDelegationType**`( int` **delegationType** `)`
- *setPolicyType*
  `public void` **setPolicyType**`( java.lang.String` **policyType** `)`
- *setProxyLifetime*
  `public void` **setProxyLifetime**`( int` **proxyLifetime** `)`
- *setProxyOutputFile*
  `public void` **setProxyOutputFile**`( java.lang.String` **proxyOutputFile** `)`
- *setProxyType*
  `public void` **setProxyType**`( int` **proxyType** `)`
- *validateACs*
  `public void` **validateACs**`( java.util.List` **ACs** `)`

16.47 CLASS VOMSREQUESTFACTORY

This class builds VOMS XML requests starting from VOMSRequestOptions objects.

16.47.1 DECLARATION

public class VOMSRequestFactory

extends java.lang.Object

16.47.2 METHODS

- *buildRequest*
  `public Document` **buildRequest**`( org.glite.voms.contact.VOMSRequestOptions` **options** `)`
- *buildRequest*
  `public String` **buildRESTRequest**`( org.glite.voms.contact.VOMSRequestOptions` **options** `)`
- *getLifetime*
  `public long` **getLifetime**`( )`
- *getOrderString*
  `public String` **getOrderString**`( )`
- *getTargetString*
  `public String` **getTargetString**`( )`
- *instance*
  `public static VOMSRequestFactory` **instance**`( )`
- *setLifetime*
  `public void` **setLifetime**`( long` **lifetime** `)`
- *setOrderString*
  `public void` **setOrderString**`( java.lang.String` **orderString** `)`
- *setTargetString*
  `public void` **setTargetString**`( java.lang.String` **targetString** `)`

16.48 CLASS VOMSREQUESTOPTIONS

This class represents options that constitute VOMS requests.

16.48.1 DECLARATION

public class VOMSRequestOptions

extends java.lang.Object

16.48.2 FIELDS

- public static final int DEFAULT_LIFETIME
  - The default lifetime value for voms requests is 86400 seconds. This default value is used if no lifetime is used with the #setLifetime(int) method.

16.48.3 CONSTRUCTORS

- *VOMSRequestOptions*
  `public` **VOMSRequestOptions**`( )`

16.48.4 METHODS

- *addFQAN*
  `public void` **addFQAN**`( java.lang.String` **FQAN** `)`
  - **Usage**
    - Adds a FQAN to the list of requested FQANs. See #getRequestedFQANs() .
  - **Parameters**
    - `FQAN` -
- *addTarget*
  `public void` **addTarget**`( java.lang.String` **target** `)`
  - **Usage**
    - Adds a target to the list of targets for this VOMSRequestOptions object. See #getTargets() .
  - **Parameters**
    - `target` -
- *getLifetime*
  `public int` **getLifetime**`( )`
  - **Returns** - the lifetime set for this VOMSRequestOptions object.
- *getOrdering*
  `public String` **getOrdering**`( )`
  - **Usage**
    - Returns the ordering string of this VOMSRequestOptions object.
  - **Returns** -
- *getRequestedFQANs*
  `public List` **getRequestedFQANs**`( )`
  - **Usage**
    - Returs the list of the requested FQANs specified in this VOMSRequestOptions object.
  - **Returns** -

- *getTargets*
  `public List` **getTargets**`( )`
  - **Usage**
    - Returns the list of targets (i.e., host where the requested ACs will be valid) for this VOMSRequestOptions object.
  - **Returns** -
- *getTargetsAsString*
  `public String` **getTargetsAsString**`( )`
  - **Usage**
    - Returns the list of targets (i.e., host where the requested ACs will be valid) for this VOMSRequestOptions object as a string containing a a comma-separated list of host names.
  - **Returns** -
- *getVerificationType*
  `public int` **getVerificationType**`( )`
- *getVoName*
  `public String` **getVoName**`( )`
- *setLifetime*
  `public void` **setLifetime**`( int` **lifetime** `)`
  - **Usage**
    - Sets the lifetime for this VOMSRequestOptions object.
  - **Parameters**
    - `lifetime` -
- *setOrdering*
  `public void` **setOrdering**`( java.lang.String` **ordering** `)`
  - **Usage**
    - Sets the ordering string of this VOMSRequestOptions object. The ordering string is used to request a spefic order for the ACs requested from the VOMS server.
  - **Parameters**
    - `ordering` -
- *setRequestedFQANs*
  `public void` **setRequestedFQANs**`( java.util.List` **requestedFQANs** `)`
  - **Usage**
    - Sets the list of requested FQANs for this VOMSRequestOptions object.
  - **Parameters**
    - `requestedFQANs` -
- *setTargets*
  `public void` **setTargets**`( java.util.List` **targets** `)`

- **Usage**
  - Sets the list of targets (i.e., host where the requested ACs will be valid) for this VOMSRequestOptions object.
- **Parameters**
  - `targets` -

- *setVerificationType*
  `public void` **setVerificationType**`( int` **verificationType** `)`
- *setVoName*
  `public void` **setVoName**`( java.lang.String` **voName** `)`

16.49 CLASS VOMSRESPONSE

This class is used to parse and represent VOMS server responses.

16.49.1 DECLARATION

public class VOMSResponse

extends java.lang.Object

16.49.2 CONSTRUCTORS

- *VOMSResponse*
  `public` **VOMSResponse**`( org.w3c.dom.Document` **res** `)`
  - **Usage**
    - Builds a VOMSResponse starting from a DOM an XML document (see Document).
  - **Parameters**
    - `res` -

16.49.3 METHODS

- *errorMessages*
  `public VOMSErrorMessage` **errorMessages**`( )`
  - **Usage**
    - Extracts the error messages from the VOMS response.
  - **Returns** - an array of VOMSErrorMessage objects.

- *getAC*
  `public byte` **getAC**`( )`
  - **Usage**
    - Extracts the AC from the VOMS response.
  - **Returns** - an array of bytes containing the AC.

- *getACAsString*
  `public String` **getACAsString**`( )`

  - **Usage**
    - Extracts the AC from the VOMS response.
  - **Returns** - a string containing the AC.

- *getVersion*
  `public int` **getVersion**`( )`

  - **Usage**
    - Extracts the version from the VOMS response.
  - **Returns** - an integer containing the AC.

- *hasErrors*
  `public boolean` **hasErrors**`( )`

16.50 CLASS VOMSSERVERINFO

This class represents information about a remote voms server as found in vomses configuration files. See VOMSESFileParser .

16.50.1 DECLARATION

public class VOMSServerInfo

extends java.lang.Object

16.50.2 CONSTRUCTORS

- *VOMSServerInfo*
  `public` **VOMSServerInfo**`( )`

16.50.3 METHODS

- *compactString*
  `public String` **compactString**`( )`
- *equals*
  `public boolean` **equals**`( java.lang.Object` **obj** `)`
- *fromStringArray*
  `public static VOMSServerInfo` **fromStringArray**`( java.lang.String [] ` **tokens** `)`
- *getGlobusVersion*
  `public String` **getGlobusVersion**`( )`
- *getGlobusVersionAsInt*
  `public int` **getGlobusVersionAsInt**`( )`
- *getHostDn*
  `public String` **getHostDn**`( )`
- *getHostName*
  `public String` **getHostName**`( )`

- *getPort*
  `public int` **getPort**`( )`
- *getVoName*
  `public String` **getVoName**`( )`
- *hashCode*
  `public int` **hashCode**`( )`
- *setGlobusVersion*
  `public void` **setGlobusVersion**`( java.lang.String` **globusVersion** `)`
- *setHostDn*
  `public void` **setHostDn**`( java.lang.String` **hostDn** `)`
- *setHostName*
  `public void` **setHostName**`( java.lang.String` **hostname** `)`
- *setPort*
  `public void` **setPort**`( int` **port** `)`
- *setVoName*
  `public void` **setVoName**`( java.lang.String` **voName** `)`
- *toString*
  `public String` **toString**`( )`

16.51 CLASS VOMSSERVERMAP

A VOMSServerMap organizes voms servers found in vomses configuration files in map keyed by vo. This way is easy to know which servers acts as replicas for the same vos. For more info about vomses configuration files, see VOMSESFileParser .

16.51.1 DECLARATION

public class VOMSServerMap

extends java.lang.Object

16.51.2 CONSTRUCTORS

- *VOMSServerMap*
  `public` **VOMSServerMap**`( )`

16.51.3 METHODS

- *add*
  `public void` **add**`( org.glite.voms.contact.VOMSServerInfo` **info** `)`
- *get*
  `public Set` **get**`( java.lang.String` **nick** `)`
- *merge*
  `public void` **merge**`( org.glite.voms.contact.VOMSServerMap` **other** `)`

- **Usage**
  - Merge this map with another VOMSServerMap object.
- **Parameters**
  - other -

- *serverCount*
  `public int` **serverCount**`( java.lang.String` **nick** `)`
- *toString*
  `public String` **toString**`( )`

16.52 CLASS VOMSSOCKET

The VOMSSocket class is used to manage the creation of the gsi socket used for communication with the VOMS server.

16.52.1 DECLARATION

public class VOMSSocket

extends java.lang.Object

16.52.2 METHODS

- *close*
  `public void` **close**`( )`
- *getContext*
  `public GSSContext` **getContext**`( )`
- *getInputStream*
  `public InputStream` **getInputStream**`( )`
- *getOutputStream*
  `public OutputStream` **getOutputStream**`( )`
- *instance*
  `public static VOMSSocket` **instance**`( org.glite.voms.contact.UserCredentials` **cred**, `java.lang.String` **hostDN** `)`
- *instance*
  `public static VOMSSocket` **instance**`( org.glite.voms.contact.UserCredentials` **cred**, `java.lang.String` **hostDN**, `int` **proxyType** `)`
- *isClosed*
  `public boolean` **isClosed**`( )`
- *isConnected*
  `public boolean` **isConnected**`( )`
- *shutdownInput*
  `public void` **shutdownInput**`( )`
- *shutdownOutput*
  `public void` **shutdownOutput**`( )`

16.53 CLASS VOMSSYNTAXEXCEPTION

16.53.1 DECLARATION

public class VOMSSyntaxException

extends org.glite.voms.contact.VOMSException

16.53.2 CONSTRUCTORS

- *VOMSSyntaxException*
  `public` **VOMSSyntaxException**`( java.lang.String` **message** `)`
- *VOMSSyntaxException*
  `public` **VOMSSyntaxException**`( java.lang.String` **message**`, java.lang.Throwable` **t** `)`
- *VOMSSyntaxException*
  `public` **VOMSSyntaxException**`( java.lang.Throwable` **t** `)`

16.53.3 METHODS INHERITED FROM CLASS `org.glite.voms.contact.VOMSException`

16.53.4 METHODS INHERITED FROM CLASS `java.lang.RuntimeException`

16.53.5 METHODS INHERITED FROM CLASS `java.lang.Exception`

16.53.6 METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *fillInStackTrace*
  `public synchronized native Throwable` **fillInStackTrace**`( )`
- *getCause*
  `public Throwable` **getCause**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *getStackTrace*
  `public StackTraceElement` **getStackTrace**`( )`
- *initCause*
  `public synchronized Throwable` **initCause**`( java.lang.Throwable` **arg0** `)`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream` **arg0** `)`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter` **arg0** `)`
- *setStackTrace*
  `public void` **setStackTrace**`( java.lang.StackTraceElement [] arg0 )`
- *toString*
  `public String` **toString**`( )`

17 KNOWN PROBLEMS AND CAVEATS

There is a set of known problems that may appear during the use of VOMS, but most of them can be eliminated by paying attention to some simple rule.

1. Remember that to obtain a VOMS-enabled proxy you *must* specify the `-voms <vo>` option. Without it voms-proxy-init generates a completely standard Globus proxy.
2. Due to the needs of GSI authentication, a maximum of 5 minutes of time shift are allowed among the VOMS server and all its clients. Any greater amount will result in a failed authentication and consequently in an error message.

REFERENCES

[1] S. Farrell, R. Housley, RFC 3281: An Internet Attribute Certificate Profile for Authorization.

[2] R. Housley, W. Polk, W. Ford, D. Solo, RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.

[3] S. Bradner, RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.

[4] V. Ciaschini, A. Frohner, Voms Credential Format, http://edg-wp2.web.cern.ch/edg-wp2/security/voms/edg-voms-credential.pdf

[5] R. Alfieri, R. Cecchini, V. Ciaschini, L. Dell'Agnello, A. Frohner, A. Gianoli, L. Karoly, F. Spataro, An Authorization System for Virtual Organizations, Forthcoming in Proceedings of the 1st European Across Grids Conference.

[6] V. Ciaschini, V. Venturi, A. Ceccanti, The VOMS Attribute Certificate Format, OGSA Authorization Working Group draft, http://forge.gridforum.org/sf/go/doc13797