EUROPEAN MIDDLEWARE INITIATIVE

EUROPEAN

MIDDLEWARE INITIATIVE

ARGUS SYSTEM ADMINISTRATOR GUIDE

Document Version:	**1.1.0**
EMI Component Version:	**1.5, 1.6**
Date:	**20.02.2013**

https://twiki.cern.ch/twiki/bin/view/EMI/ArgusEMIDocumentation

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

Functional Description

The Argus Authorization Service renders consistent authorization decisions for distributed services (e.g., user interfaces, portals, computing elements, storage elements). The service is based on the XACML standard, and uses authorization policies to determine if a user is allowed or denied to perform a certain action on a particular service.

The Argus Authorization Service is composed of three main components:

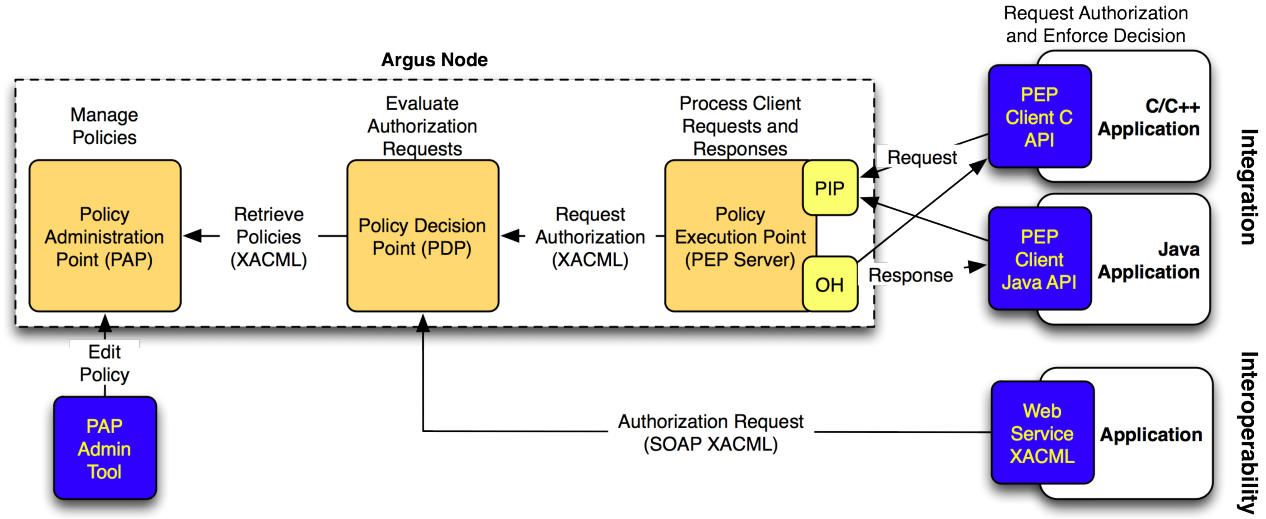
Policy Administration Point (PAP)
: Provides the tools to author authorization policies, organize them in the local repository and configure policy distribution among remote PAPs.

Policy Decision Point (PDP)
: Implements the authorization engine, and is responsible for the evaluation of the authorization requests against the XACML policies retrieved from the PAP.

Policy Enforcement Point Server (PEP Server)
: Handles the authorization requests received from the PEP client, and ensures the integrity and consistency. Configurable policy information points (PIP) can transform or complete the incoming requests. Oobligation handlers (OH) can be applied to the resulting response.

Policy Enforcement Point client (PEP client)
: Lightweight PEP client libraries are also provided to ease the integration and interoperability with other EMI services or components.

The following graphic shows the interaction between the components of the service:



Note: In Argus, the PEP is separated in a client/server architecture. The PEP Server handles the lightweight PEP client requests, and runs on the Argus node.

Please have a look to the Argus Authorization Service twiki for more information and documentation

System Administrator Documentation

- System Administrator Guide:
  - PDF: EMI Argus System Administrator Guide 1.1.0 (pdf)
  - Argus Service Deployment: https://twiki.cern.ch/twiki/bin/view/EGEE/ArgusEMIDeployment
  - Argus Authorization Service: https://twiki.cern.ch/twiki/bin/view/EGEE/AuthorizationFramework

- Service Reference Card:
  - PDF: EMI Argus Service Reference Card 1.1.0 (pdf)
  - Argus Service Reference Card: https://twiki.cern.ch/twiki/bin/view/EMI/ArgusSRC

Argus Service Deployment for EMI

Argus Service Reference Card: https://twiki.cern.ch/twiki/bin/view/EMI/ArgusSRC

Requirements

Platform

- For **EMI-3** the supported platforms are **Debian6/x86_64**, and **SL5/x86_64** and **SL6/x86_64** with the EPEL repository **enabled**.
- For **EMI-2** the supported platforms are **SL5/x86_64** and **SL6/x86_64** with the EPEL repository **enabled**.
- For **EMI-1** the supported platform is **SL5/x86_64** with the EPEL repository **enabled**.

Host Certificate

The Argus services require a valid host certificate. The key pair is typically installed in `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem`

Installation with YUM

The Argus Authorization Service is a bundle of 3 services (PAP, PDP and PEP Server) and is available in the EMI repository. The Argus metapackage should be installed with YUM.

Install the EGI IGTF Bundle

If not already present on your host, install the EGI IGFT trust anchors. More information available online: https://wiki.egi.eu/wiki/EGI_IGTF_Release

- Install the EGI IGTF trust anchors with YUM
- Install fetch-crl (from the EPEL repository), run it once, and enable the cron job:

```
# install
yum install fetch-crl
# run it immediately (it can take some time...)
/usr/sbin/fetch-crl
# enable the periodic fetch-crl cron job
/sbin/chkconfig fetch-crl-cron on
/sbin/service fetch-crl-cron start
```

Install the EMI Repository

If not already configured on your host, install the EMI repositories package as described in the EMI generic installation and configuration guide for EMI 1 or EMI 2, or EMI 3.

And update the YUM cache: `yum makecache`

Install the Argus Metapackage

The `emi-argus` metapackage bundles the 3 Argus services: PAP, PDP and PEP Server

Therefore, the `emi-argus` metapackage is the simplest way to install the Argus Authorization Services

(PAP, PDP and PEP Server) on your host.

Use YUM to install the Argus metapackage: `yum install emi-argus`

At this point all 3 Argus services (PAP, PDP and PEP server) are installed on your host. You must now continue with the Argus service configuration with YAIM.

Update with YUM

Argus and its components are packages in the EMI and UMD repositories, and therefore automatically updated if a new version is available and `yum update` is executed. The updating process however is stopping the Argus services (PAP, PEPd and PDP). To get Argus back to work after an update it is recommended to rerun Yaim and/or restart the services.

Please make sure that your system does not use the automated yum updating service, since this may lead to a stopped Argus server in case of a unnoticed update.

Configuration with YAIM

The `ARGUS_server` node type is available to configure the Argus service with YAIM.

Argus YAIM Configuration Variables

Description of all the available Argus YAIM configuration variables:
https://twiki.cern.ch/twiki/bin/view/EGEE/ArgusEMIYaimConfiguration

Argus `site-info.def` Configuration

Your `site-info.def` for Argus must contain at least the following variable:

```
# BDII site name
SITE_NAME=MySiteName

# Argus service hostname
ARGUS_HOST=argus.example.org

# PAP administrator DN allowed to use 'pap-admin' command
PAP_ADMIN_DN="/DC=org/DC=acme/CN=John Doe"

# Users and Groups definition for grid and group mapfile
USERS_CONF=/opt/glite/yaim/examples/users.conf
GROUPS_CONF=/opt/glite/yaim/examples/groups.conf

# Supported VOs
VOS="dteam"

VO_DTEAM_VOMSES="'dteam voms.hellasgrid.gr 15004 /C=GR/O=HellasGrid/OU=hellasgrid.gr/
CN=voms.hellasgrid.gr dteam' 'dteam voms2.hellasgrid.gr 15004 /C=GR/O=HellasGrid/
OU=hellasgrid.gr/CN=voms2.hellasgrid.gr dteam'" VO_DTEAM_VOMS_CA_DN="'/C=GR/O=HellasGrid/
OU=Certification Authorities/CN=HellasGrid CA 2006' '/C=GR/O=HellasGrid/OU=Certification
Authorities/CN=HellasGrid CA 2006'"
```

Documentation of all the Argus YAIM Configuration Variables.

Generate Argus Configuration

Run YAIM to generate the Argus configuration for your site:

```
/opt/glite/yaim/bin/yaim -c -s site-info.def -n ARGUS_server
```

Install the Argus Metapackage

At this point, the Argus services (PAP, PDP and PEP Server) must be configured, up and running.

Nagios Probes for Argus

A set of Nagios probes for Argus (EMI-2 and EMI-3) are available to monitor the Argus PAP, PDP and PEP Server:

- Argus Nagios Probes Documentation (EMI):
  https://twiki.cern.ch/twiki/bin/view/EGEE/ArgusEMINagiosProbes

EMIR Publisher for Argus 1.6 (EMI-3)

You can use EMIR-SERP to publish the Argus resource information to EMIR. EMIR-SERP uses the information already available in the resource BDII and publish it to an EMIR DSR endpoint.

- See the Argus EMIR Configuration to publish the Argus into EMIR: https://twiki.cern.ch/twiki/bin/view/EGEE/ArgusEMIEmirConfiguration

Known Issues

Problem with Argus 1.5 (EMI-2) and CREAM

Under heavy load the Argus PEP Server (v1.5.1) does not always return a user mapping for a permitted operation, causing CREAM to throw an error and abort the job. This typically occurs for 10% of the jobs submitted by CREAM.

Workaround

Disabling the PDP responses caching mechanism in the PEP Server configuration solve this issue. To disable the cache:

1. edit `/etc/argus/pepd/pepd.ini`
2. add the parameter `maximumCachedResponses = 0` in the `[PDP]` section (see example below)
3. restart PEP Server: `/etc/init.d/argus-pepd restart`

Example:

```
[PDP]
pdps = https://chaos.switch.ch:8152/authz
# disabling the cache
maximumCachedResponses = 0
```

Problem with upgrade from Argus 1.4 (EMI-1) to Argus 1.5 (EMI-2)

When upgrading an previous Argus 1.4 (EMI-1) installation on SL5, you need to re-install Argus 1.5 (EMI-2). This is due to an error in the Argus 1.4 post uninstall script.

Workaround

Simply reinstalling the components with YUM just after the upgrade solves the issue:

```
yum upgrade
(argus is upgraded...)
yum reinstall argus-pap argus-pdp argus-pep-server
```

Argus YAIM Configuration for EMI

YAIM Configuration for ARGUS_server

Mandatory General Variables

- SITE_NAME BDII site name
- USERS_CONF
- GROUPS_CONF
- VOS List of supported VO names
- VO_<vo-name>_VOMS_CA_DN VOMS CA DN for each VO name listed in VOS
- VO_<vo-name>_VOMSES VOMS definition for each VO name listed in VOS

More information on these variables available here:
https://twiki.cern.ch/twiki/bin/view/LCG/Site-info_configuration_variables

Mandatory Service Specific Variables

They can be found in
/opt/glite/yaim/examples/siteinfo/services/glite-argus_server

Variable Name	Description	Value type	Version
ARGUS_HOST	Hostname of the Argus node.	FQDN Hostname	1.1.0-1
PAP_ADMIN_DN	User certificate DN of the user that will be the PAP administrator.	Certificate DN	1.0.0-1

Default Service Specific Variables

They can be found in /opt/glite/yaim/defaults/glite-argus_server(.pre|.post)

Variable Name	Description	Value type	
CONFIG_PAP	Set this variable to no if you don't want yaim to create the PAP configuration files	string	yes
CONFIG_PDP	Set this variable to no if you don't want yaim to create the PDP configuration file	string	yes
CONFIG_PEP	Set this variable to no if you don't want yaim to create the PEP Server configuration file	string	yes
PAP_HOME	Home directory of the pap service	path	${PAP_HOME:-"/usr/share/argus
PAP_ENTITY_ID	This is a unique identifier for the PAP. It must be a URI (URL or URN) and the same entity ID	URI	${PAP_ENTITY_ID:-"http://${AR

	should be used for all PAP instances that make up a single logical PAP. If a URL is used it doesn't neet to resolve to any specific webpage.		
PAP_HOST	Set this variable to another value if PAP_HOST is not installed in the same host as PDP and PEP.	IP/DNS name	${ARGUS_HOST}
PAP_CONF_INI	Configuration file for the pap service	path	${PAP_CONF_INI:-"${PAP_HOME}/
PAP_AUTHZ_INI	Configuration file for the pap service authorization policies	path	${PAP_AUTHZ_INI:-"${PAP_HOME}
PAP_ADMIN_PROPS	Configuration properties for the pap-admin client	path	${PAP_ADMIN_PROPS:-"${PAP_HOM
PAP_REPO_LOCATION	Path to the repository directory	path	${PAP_REPO_LOCATION:-"${PAP_H
PAP_POLL_INTERVAL	The polling interval (in seconds) for retrieving remote policies	number	14400
PAP_ORDERING	Comma separated list of pap aliases. Example: alias-1, alias-2, ..., alias-n. Defines the order of evaluation of the policies of the paps, that means that the policies of pap "alias-1" are evaluated for first, then the policies of pap "alias-2" and so on.	string	default
PAP_CONSISTENCY_CHECK	Forces a consistency check of the repository at startup.	boolean	false
PAP_CONSISTENCY_CHECK_REPAIR	if set to true automatically fixes problems detected by the consistency check (usually means deleting the corrupted policies).	boolean	false
PAP_PORT	PAP standalone service port	port	8150
PAP_SHUTDOWN_PORT	PAP standalone shutdown service port	port	8151

Default Service Specific Variables

PAP_SHUTDOWN_COMMAND	PAP standalone shutdown command (password)	port	generated pseudo random
PDP_HOME	Home directory of the pdp service	path	${PDP_HOME:-"/usr/share/argus
PDP_CONF_INI	Configuration file for the PDP service	path	${PDP_CONF_INI:-"/etc/argus/p
PDP_ENTITY_ID	This is a unique identifier for the PEP. It must be a URI (URL or URN) and the same entity ID should be used for all PEP instances that make up a single logical PEP. If a URL is used it need not resolve to any specific webpage.	URI	${PDP_ENTITY_ID:-"http://${AR
PDP_HOST	Set this variable to another value if PDP_HOST is not installed in the same host as PAP and PEP.	IP/DNS name	${ARGUS_HOST}
PDP_PORT	PDP standalone service port	port	8152
PDP_ADMIN_PORT	PDP admin service port	port	8153
PDP_ADMIN_PASSWORD	PDP admin service password for shutdown, reload policy, ..., commands	port	generated pseudo random
PDP_RETENTION_INTERVAL	The number of minutes the PDP will retain (cache) a policy retrieved from the PAP. After this time is passed the PDP will again call out to the PAP and retrieve the policy	number	240
PDP_PAP_ENDPOINTS	Space separated list of PAP endpoint URLs for the PDP to use. Endpoints will be tried in turn until one returns a successful response. This provides limited failover support. If more intelligent failover is necessary or load balancing is required, a dedicated	URLs	${PDP_PAP_ENDPOINTS:-"https:/

	load-balancer/failover appliance should be used.		
PEP_HOME	Home directory for the pep service	path	${PEP_HOME:-"/usr/share/argus
PEP_CONF_INI	Configuration for the pep service	path	${PEP_CONF_INI:-"/etc/argus/p
PEP_ENTITY_ID	This is a unique identifier for the PEP. It must be a URI (URL or URN) and the same entity ID should be used for all PEP instances that make up a single logical PEP. If a URL is used it need not resolve to any specific webpage.	URI	${PEP_ENTITY_ID:-"http://${AR
PEP_HOST	Set this variable to another value if PEP_HOST is not installed in the same host as PAP and PDP. But remember to use the hostname and not 127.0.0.1 !	IP/DNS name	${ARGUS_HOST}
PEP_PORT	PEP service port	port	8154
PEP_ADMIN_PORT	PEP admin service port	port	8155
PEP_ADMIN_PASSWORD	PEP admin service password for shutdown, clear cache, ..., commands	port	generated pseudo random
PEP_MAX_CACHEDRESP	The maximum number of responses from any PDP that will be cached. Setting this value to 0 (zero) will disable caching.	number	500
PEP_PDP_ENDPOINTS	Space separated list of PDP endpoint URLs for the PEP to use. Endpoints will be tried in turn until one returns a successful response. This provides limited failover support. If more intelligent failover is necessary or load balancing is required, a dedicated load-balancer/failover	URLs	${PEP_PDP_ENDPOINTS:-"https:/

	appliance should be used.		

Argus Policy Administration Point (PAP): Configuration

Configuration Files

The PAP is configured through the use of two files: `pap_configuration.ini` and `pap_authorization.ini`, located in the `/etc/argus/pap` (EMI), or `$PAP_HOME/conf` (gLite), directory. Most of the information contained in these files can also be set through the command line interface (which is the **recommended** way to do configuration on the PAP).

Service configuration file

The service is primarily configured through the `pap_configuration.ini` configuration file. This file is a standard INI file with five defined sections.

Section: [paps]

This section contains configuration about paps. The information in this section should be set via the command line interface (see the #Pap_management_commands commands group).

A paps can be defined by providing the following information (the R value in the Required? column indicates information that is required only for remote paps):

Property	Description	Required?	Default Value
alias.type	Defines a pap as `local` or `remote` .	Y	None
alias.public	Visibility of the pap: `true` or `false`. If false its policies are not sent to other paps.	N	false
alias.enabled	`true` or `false`. If false its policies are not sent to PDPs.	N	false
alias.dn	DN of the PAP to get policies from.	R	None
alias.hostname	Hostname of the PAP to get policies from.	R	None
alias.port	Port of the PAP to get policies from.	N	8150
alias.path	Path of the services exposed by the PAP to get policies from.	N	/pap/services
alias.protocol	Protocol to use to contact the remote PAP.	N	https

Section: [paps:properties]

This section contains information about policy distribution and pap ordering.

Property	Description	Required?	Default Value
poll_interval	The polling interval (in seconds) for retrieving remote policies.	Y	None. Recommended value is 14400 (4 hours).
ordering	Comma separated list of pap aliases. Example: alias-1, alias-2, ..., alias-n. Defines the order of evaluation of the policies of the paps, that means that the policies of pap "alias-1" are evaluated for first, then the policies of pap "alias-2" and so on.	N	If not specified the `default` pap is always the first one.

Section: [repository]

This section contains information about the PAP policy repository.

Property	Description	Required?	Default Value
location	Path to the repository directory.	N	$PAP_HOME/repository
consistency_check	Forces a consistency check of the repository at startup.	N	false
consistency_check.repair	if set to *true* automatically fixes problems detected by the consistency check (usually means deleting the corrupted policies).	N	false

Section: [standalone-service]

This section contains information about the PAP standalone service.

Property	Description	Required?	Default Value
hostname	The hostname or IP address the service will bind to	N	127.0.0.1
port	The service port number	N	8150
shutdown_port	The service shutdown port number	N	8151
shutdown_command	The command string that must be received on the shutdown port in order to shutdown the service. The command is needed in order to prevent unauthorized shutdown commands coming from localhost. This is effective only if the pap_configuration.ini file is not world-readable. If the option is not present in configuration, no check on the command will be made.	N	shutdown
entity_id	This is a unique identifier for the PAP. It must be a URI (URL or URN). If a URL is used it need not resolve to any specific webpage.	N	The service endpoint, e.g. `https://pap.cern.ch:8150/pap/services/Pro`

Section: [security]

This sections contains information about PAP security configuration.

Property	Description	Required?	Default Value
certificate	The X.509 pem-econded service certificate	Y	/etc/grid-security/hostcert.pem
private_key	The unencrypted private key bound to the certificate	Y	/etc/grid-security/hostkey.pem
trust_store_dir	The directory where CA files and CRL are looked for	N	/etc/grid-security/certificates
crl_update_interval	How frequently the PAP should update CRLs, CAs and namespaces from the filesystem. The interval is defined as a string with the following format: `N{s,m,h,d}` where N in the number of either (s=seconds, m=minutes, h=hours, d=days).	N	30m

Service Access Control

Access control rules are configured through the `pap_authorization.ini` configuration file. Authorization is based on off of the Subject DN or VOMS attribute within the client certificate used to authenticate to the PAP.

PAP permissions

The authorization layer is based on an Access Control List (ACL), composed of several Access Control Entries (ACEs). Each ACE defines the actions that an administrator is allowed to execute on the PAP. Administrators' privileges are defined in terms of PAP permission flags, whose meaning is described in the table below:

Permission Flag	Meaning
`POLICY_READ_LOCAL`	Allows read access to locally defined policies
`POLICY_READ_REMOTE`	Allows read access to policies imported from remote PAPs
`POLICY_WRITE`	Allows write access to locally defined policies
`CONFIGURATION_READ`	Allows read access to PAP configuration
`CONFIGURATION_WRITE`	Allows write access to PAP configuration
`ALL`	All of the above permissions

A set of permission flags can be assigned to an administrator by defining an ACE in the `pap_authorization.ini` configuration file or by using the authorization management commands provided by the `pap-admin` command line interface.

ACEs are expressed as

```
<principal> : <permission>
```

couples.

The `principal` part of the ACE is either:

- `ANYONE`, to assign privileges to any authenticated user (i.e., any user that presents a trusted certificate).
- a VOMS FQAN, e.g., `/atlas/Role=VO-Admin`

- a **quoted** X509 certificate subject, e.g., `"/C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Andrea Ceccanti"`

The `permission` part of the ACE is either:

- a single PAP permission flag, e.g `CONFIGURATION_READ`
- a `|` separated list of PAP permission flags, e.g. `POLICY_READ_LOCAL|CONFIGURATION_READ`, to grant a set of permissions.

So, for example, to grant `POLICY_READ_LOCAL` and `POLICY_READ_REMOTE` permissions to a user identified by an x509 certificate with `/C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Andrea Ceccanti` subject, one should write:

```
"/C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Andrea Ceccanti" : POLICY_READ_LOCAL|POLICY_READ_
```

Note that the subject has been put into quotes! For VOMS FQANs this is not needed (FQAN syntax does not allow whitespaces inside the FQAN), so one could write:

```
/atlas/Role=PAP-Admin : ALL
```

Authorization entries are loaded at PAP service startup time so any `pap_authorization.ini` modifications done "by hand" while the PAP service is running do not take effect until the PAP service is restarted.

To modify the PAP authorization configuration at runtime, use the authorization management commands provided by the `pap-admin` CLI. Changes made to the PAP ACL by these commands are immediately reflected on the `pap_authorization.ini` file.

Configuration File Syntax

In the `pap_authorization.ini` file, ACEs are grouped in two *stanzas* according to the type of the principal. Currently, two stanzas are supported:

- `[dn]`, that lists ACEs defined for principals identified by an X509 certificate subject.
- `[fqan]`, that lists ACEs defined for principals identified by VOMS fqans.

An example of configuration file is given below:

```
[dn]

"/C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Andrea Ceccanti" : ALL

ANYONE : CONFIGURATION_READ|CONFIGURATION_WRITE

[fqan]

/voms-ws/Role=PAP-Admin : ALL
```

Argus Policy Administration Point (PAP): Administration

The CLI (Command Line Interface) allows to perform all of the policy management operations as well as to set most of the configuration information of the PAP including authorization settings. All these operations are accessible by means of (sub)commands of the PAP CLI and they are divided in three sections: *policy management*, *pap management* and *authorization management*.

The command to invoke the CLI is `pap-admin`.

Running the pap-admin client

Type:

```
pap-admin --help
```

to get a list of the commands supported by the current version of the PAP command line client.

The general usage is the following:

```
usage: pap-admin [global-options] <command> [options] [args]
```

- `global-options` : are options shared by all commands (all these options are reported by `--help`).
- `command` : the command to perform.
- `options` : command specific options.

Type:

```
pap-admin <command> --help
```

to see the available command specific options.

Global options:

- `-h, --help`: help message;
- `-v, --verbose`: set verbose commands output;
- `--host <arg>`: specifies the target PAP hostname (default is localhost). This option defines the PAP endpoint to be contacted as follows: https://arg:port/pap/services/;
- `--port <arg>`: specifies the port on which the target PAP is listening (default is 8150);
- `--url <arg>`: specifies the full target PAP endpoint (default: https://localhost:8150/pap/services/);
- `--cert <arg>`: specifies non-standard user certificate;
- `--key <arg>`: specifies non-standard user private key;
- `--password <arg>`: specifies the password used to decrypt the user's private key;
- `--proxy <arg>`: specifies a user proxy to be used for authentication;

By default, the `pap-admin` is configured to contact a PAP daemon on localhost port 8150 (i.e., the standalone PAP daemon). If you want to contact a PAP on another host use the `--host` option, a PAP listening on a different port use the `--port` option. Alternatively the full endpoint can be specified with the `--url` option.

NOTE: The Argus PDP caches the policy pulled from the PAP. If you make policy changes to the PAP you will need to restart the PDP in order to force it to refresh its policy. Also, the PEPd caches results from the

PDP for a short time (up to 10 minutes, by default) so you may need to restart the PEPd as well.

Exit status

Each command returns 0 for success or non-zero in case of error.

X509 authentication

A valid X509 certificate or proxy certificate is needed in order to run the `pap-admin` client. The certificate to be used by the command is found as follows:

1. if the user is *root* then the client looks for host certificate at the usual location, i.e. `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem`. These locations can be overridden by setting the `X509_USER_CERT` and `X509_USER_KEY` environment variables.
2. if the user is not *root*, then the client looks for a proxy certificate at the usual location, i.e., `/tmp/x509up_u<uid>`. The default proxy location can be overridden using the `X509_USER_PROXY` environment variable. If no proxy is found, then the client looks for a certificate (and the relative private key) at the usual location, i.e., `$HOME/.globus/usercert.pem` and `$HOME/.globus/userkey.pem`. These locations can be overriden by setting the `X509_USER_CERT` and `X509_USER_KEY` environment variables.

This behaviour can be overridden by specifying the `--cert`, `--key` and `--proxy` command line options.

Client configuration (since version 1.3.0)

The `pap-admin.properties` file, found in `/etc/argus/pap` (EMI) or `/opt/argus/pap/conf` (gLite), allows to set the following properties:

Property	Description	Default Value
host	The PAP host that will be contacted by the pap-admin CLI if no host is explicitly specified with the `--host` option	localhost
port	The remote PAP service port number	8150

This property file is expecially useful when administering a PAP installed on a remote machine.

Policy Management Commands

This set of commands allows to perform policy management operations. All the commands in this section modifies the *default* pap. In order to target another local pap use the option `--pap <alias>`.

Command: list-policies

List policies. By default the policies of the *default* pap are listed unless option `--pap` is specified.

```
usage: pap-admin [global-options] list-policies [options]
```

Command specific options:

- `--pap <alias>`: list policies of pap "alias" (*default* pap is assumed if this option is missing);
- `--all`: list policies of all the defined paps;
- `-srai,--show-ra-ids`: show resource and action ids;
- `-sai,--show-all-ids`: show all ids (resource, action and rule ids);

Running the pap-admin client

- `--show-xacml`: print policies using XACML.

Command: ban

Allows to ban an attribute (i.e. SUBJECT, FQAN, CA, etc.).

A *deny* rule is added for the given attribute into the specified resource/action value. If the resource or the action values are not specified then ".*" is assumed.

```
usage:  pap-admin [global-options] ban [options] <id> <value>
```

- `id`: id of the attribute. The list of supported id depends on the Argus version:
  - Argus v. 1.0: dn, ca, vo, fqan, pfqan.
  - Argus v. 1.1: subject, subject-issuer, vo, fqan, pfqan.

- `value`: value of the attribute - **note:** If you are using Argus v. 1.0, the DN must be in RFC2253 format, which can be obtained from `openssl` using the command `openssl x509 -in <cert.pem> -noout -subject -nameopt rfc2253`

Command specific options:

- `-a,--action <value>`: specify an action value (default is ".*")
- `-r,--resource <value>`: specify a resource value (default is ".*")
- `--pap <alias>`: add the policy to the pap "alias" (*default* pap is assumed if this option is missing)
- `--private`: set the policy as private
- `--public`: set the policy as public

Semantic of the command: the *resource* and the *action* where the deny rule is inserted are chosen as follows:

- if the first resource found in the repository matches the given one, then that resource is used, otherwise a new one is created.
- if a matching resource was found, then if its first action matches the given one then this action is used, otherwise a new action is created (i.e. inside the new resource or inside the matched resource).
- otherwise a new resource/action are created and inserted on the top.

Example:

```
pap-admin ban subject "CN=host.test.foo.it, L=FOO, OU=Host, O=ORGANIZATION, C=IT"
```

Command: un-ban

Allows to un-ban an attribute (i.e. SUBJECT, FQAN, CA, etc.), that means removing a `deny` rule (if it exists in the given resource/action) for the given attribute.

```
usage: pap-admin [global-options] un-ban [options] <id> <value>
```

- `id`: id of the attribute. The list of supported id depends on the Argus version:
  - Argus v. 1.0: dn, ca, vo, fqan, pfqan.
  - Argus v. 1.1: subject, subject-issuer, vo, fqan, pfqan.

- `value`: value of the attribute - **note:** If you are using Argus v. 1.0, the DN must be in RFC2253 format, which can be obtained from `openssl` using the command `openssl x509 -in <cert.pem> -noout -subject -nameopt rfc2253`

Command: list-policies

Command specific options:

- `-a, --action <value>`: specify an action value (default is ".*")
- `-r, --resource <value>`: specify a resource value (default is ".*")
- `--pap <alias>`: remove the ban policy from the pap *alias* (*default* pap is assumed if this option is missing)

Semantic of the command: the target *resource* and *action* to search the deny rule for are chosen as follows:

- the target *resource* is the first matching *resource* in the repository;
- inside the target *resource* the target *action* is the first matching *action* ;
- if no target *resource* or *action* were found than the result is an error message saying "ban policy not found".

Example:

```
pap-admin un-ban subject "CN=host.test.foo.it, L=FOO, OU=Host, O=ORGANIZATION, C=IT"
```

Command: add-policy

Add a permit/deny policy.

```
usage: pap-admin [global-options] add-policy [options] <permit|deny> <id=value>...
```

- `permit|deny`: effect of the policy.
- `id=value`: a string in the form "=", where *id* is any of the attribute ids that can be specified in the simplified policy language and *value* the value to be assigned (e.g. fqan=/vo/group).

Required command options:

1. `--action-id <action-id>` optionally with `--rule-id <rule-id>`: allows to specify an action-id to insert the policy into.
2. `--resource <value>` and `--action <value>`: allows to specify a resource/action value to insert the policy into.

The two groups (1 and 2) of required options are mutually exclusive.

This command allows to add a (permit/deny) rule into an action by specifying an action-id (in this case the action must already exist) or a resource/action value. In the latter case a new resource and/or action are created if they don't already exist. The command returns an error if there are more than one existing resource and/or action with the same value. By default the rule is inserted at the top of an action unless the `--bottom` option is given. If the `--rule-id` is set the rule is inserted before the given rule-id or after if the `--after` option is present.

Command specific options:

- `--pap <alias>`: add the policy to the pap "alias" (*default* pap is assumed if this option is missing);
- `--action-id <action-id>`: specify an action id;
- `--rule-id <rule-id>`: specify a rule id (requires option `--action-id`);
- `--resource <value>`: specify a resource value;
- `--action <value>`: specify an action value;
- `--after`: insert the rule after the given rule id;
- `--bottom`: insert the rule at the bottom of the list of rules of the action.
- `--obligation <obligationId>`: specify an obligation. (**since version 1.2.0**)

Command: un-ban

- `--obligation-scope <scope>`: Defines in which scope the obligation will be defined. Possible values: `action`, `resource`. If not specified, `resource` is used as default. (**since version 1.2.0**)

Command: add-policies-from-file

Add policies (resources or actions) defined in the given file.

```
usage: pap-admin [global-options] add-policies-from-file [options] <file> [resourceId]
```

- `file`: text file containing the policies to add (policies defined with the simplified policy language)
- `resourceId`: the *resource* to insert the policies into.

If *resourceId* is not specified then *file* must contain *resource* elements that will be added, by default, at the bottom (unless option `--pivot` is specified). Otherwise if *resourceId* is not specified then *file* must contain *action* elements that will be added, by default, at the bottom inside *resourceId* (unless option `--pivot` is specified).

Command specific options:

- `--pap <alias>`: add the policies to the pap "alias" (*default* pap is assumed if this option is missing);
- `--pivot <id>`: insert before ;
- `--after`: modifies the behavior of the `--pivot` option in insert after .

Command: update-policy-from-file

Update a resource/action with a new resource/action defined in a given file.

```
usage: pap-admin [global-options] update-policy-from-file [options] <id> <file>
```

- `id`: id, as listed by the command `pap-admin lp --show-all-ids` command, of the resource or action to be updated;
- `file`: text file containing the new policy definition (using the simplified policy language syntax).

In order to modify an action the `file` must contain only the new action, for example:

```
action ".*" {
    rule deny { subject="/DC=ch/DC=cern/OU=Organic  Units/OU=Users/CN=user/CN=111111/CN=user name
}
```

Command specific options:

- `--pap <alias>`: update the policies for pap "alias" (*default* pap is assumed if this option is missing);

Command: remove-policy

Remove policy by id.

```
usage: pap-admin [global-options] remove-policy [options] id...
```

- `id`: id, as listed by the command `pap-admin lp --show-all-ids` command, of the policy (resource, action or rule) to remove;

Command specific options:

Command: add-policy

- `--pap <alias>`: remove policies of pap "alias" (*default* pap is assumed if this option is missing);

Command: remove-all-policies

Remove all policies of a pap. Use option `--pap` to specify a pap different than the default one.

```
usage: pap-admin [global-options] remove-all-policies [options]
```

Command specific options:

- `--pap <alias>`: remove the policies of pap "alias" (*default* pap is assumed if this option is missing);

Command: move

Move a resource, action or rule before or after another, respectively, resource, action or rule.

```
usage: pap-admin [global-options] move [options] <id> <pivotId>
```

- `id`: id, as listed by the command `pap-admin lp --show-all-ids` command, of the policy (resource, action or rule) to move;
- `pivotId`: id of the pivot policy (*id* is moved before *pivotId*)

If *id* refers to a resource, action or rule then *pivotId* must be, respectively, a resource, action or rule id.

Command specific options:

- `--pap <alias>`: move the policy of pap "alias" (*default* pap is assumed if this option is missing);
- `--after`: move *id* after *pivotId*.

Command: add-obligation (since version 1.2.0)

Adds on obligation to an existing resource or action policy.

```
usage: pap-admin [global-options] add-obligation <policyId> <obligationId>
```

- `policyId`: the id of the policy where the obligation is to be added. In order the get the policyId of existing policies, run the `list-policies` command with the `--show-all-ids` option.
- `obligationId`: the id of the obligation that will be added.

Command specific options:

- `--pap <alias>`: add on policies defined in the pap "alias" (*default* pap is assumed if this option is missing);

Command: remove-obligation (since version 1.2.0)

Removes an obligation from an existing resource or action policy.

```
usage: pap-admin [global-options] remove-obligation <policyId> <obligationId>
```

- `policyId`: the id of the policy where the obligation is to be removed. In order the get the policyId of existing policies, run the `list-policies` command with the `--show-all-ids` option.
- `obligationId`: the id of the obligation that will be removed.

Command: remove-policy

Command specific options:

- `--pap <alias>`: add on policies defined in the pap "alias" (*default* pap is assumed if this option is missing);

PAP Management Commands

This set of commands allows to perform management operations of the PAPs.

Command: ping

Ping a PAP and return version information.

```
usage: pap-admin [global-options] ping
```

Command: add-pap

Add a remote or local pap.

```
usage: pap-admin [global-options] add-pap [options] <alias> [<endpoint> <dn>]
```

- `alias`: a friendly (unique) name used to identify the pap
- `endpoint`: endpoint of the remote pap in the form:

  ```
  [<protocol>://]<host>:[<port>/[path]]
  ```
- `dn`: DN of the remote pap

A just added pap is disabled by default (its policies are not sent to the PDP), use the command `enable-pap` to enable it (see #Command_enable_pap).

By default a pap is considered to be private (use the `--public` option to set the pap as public). Policies defined in a public pap can be fetched from other remote PAPs, while this is not allowed when the PAP is set to private.

If `endpoint` and `dn` are present the pap is considered to be remote (unless option `--local` is specified), otherwise it is local. For the endpoint the only required parameter is the hostname, these are the default values:

- *protocol*: https
- *port*: 8150
- *service path*: pap/services

When a new pap is added, the PAP service tries immediately to fetch its policies. If the remote pap is not reachable, the pap-admin command prints an error message clarifying that the pap was successfully added, but the fetching of the policies failed.

If the option `--no-policies` is given, the policies are not fetched at pap creation time but automatically by the server every `polling interval` seconds or manually when the a `refresh-cache` command is sent to the server.

Examples of endpoint are:

- `test.site.com` (hostname);
- `test.site.com:9999` (hostname and port);

Command: remove-obligation (since version 1.2.0)

- `test.site.com:9999/service_path` (hostname, port, and service path);
- `https://test.site.com:9999/service_path` (full URL).

Command specific options:

- `-l, --local`: set the pap as local;
- `--remote`: set the pap as remote;
- `--private`: set the pap as private;
- `--public`: set the pap as public;
- `--no-policies`: do not fetch the policies now.

Example:

```
pap-admin add-pap cnaf_pap test.cnaf.infn.it "/C=IT/O=INFN/OU=Host/L=CNAF/CN=test.cnaf.infn.it"
```

Command: update-pap

Update pap information.

```
usage: pap-admin [global-options] update-pap [options] <alias> [<endpoint> <dn>]
```

The input is the same as for the "add-pap" command, the effect is to update old information with the new one. The *alias* of a pap cannot be modified. In the case of a remote pap the policies are fetched immediately unless option `--no-policies` is given.

Command: remove-pap

Remove a pap and delete its policies.

```
usage: pap-admin [global-options] remove-pap <alias>
```

- `alias`: alias of the pap to remove

Command: list-paps

List all defined paps.

```
usage: pap-admin [global-options] list-paps [options]
```

Command specific options:

- `-l`: use a long list format (displays all the information of a pap).

Command: enable-pap

Set a pap as enabled (i.e. PDPs will get its policies).

```
usage: pap-admin [global-options] enable-pap <alias>
```

Command: disable-pap

Set a pap as disabled (i.e. PDPs won't get its policies).

```
usage: pap-admin [global-options] disable-pap <alias>
```

Command: add-pap

Command: get-paps-order

Get paps ordering.

```
usage: pap-admin [global-options] get-paps-order
```

If no ordering is defined the output message is: `No ordering has been defined`. If the *default* pap is not listed in the ordering (like in the no ordering defined case) by default it is placed for first.

Command: set-paps-order

Define paps ordering.

```
usage: pap-admin [global-options] set-paps-order [alias]...
```

- `alias`: a valid pap alias.

All the aliases must be valid (existing). If no arguments are given then the current ordering (if there's any defined) is deleted.

Example:
The remote pap *osct* contains banning policies and we want that policies to be evaluated for first. This is command to issue:

```
pap-admin set-paps-order osct default
```

If the PAP service contains other paps beyond the *osct*, then their policies are evaluated after the *osct* and *default* pap policies. Since the ordering contains only the *osct* and the *default* paps it is not guaranteed a special order for the evaluation of the policies of all the other paps (except that they are evaluated after these two paps).

Command: refresh-cache

Invalidates the local policy cache and retrieves policies from remote paps.

```
usage: pap-admin [global-options] refresh-cache [alias]...
```

- `alias`: a valid pap alias.

The arguments identify the paps that will be contacted. If no arguments are given, all the defined remote paps are contacted.

Command: get-polling-interval

Get the polling interval in seconds.

```
usage: pap-admin [global-options] get-polling-interval
```

Command: set-polling-interval

Invalidates the local policy cache and retrieves policies from remote paps.

```
usage: pap-admin [global-options] set-polling-interval <seconds>
```

- `seconds`: polling interval in seconds.

Authorization Management Commands

This set of commands implement Access Control List (ACL) management for PAP administrators.

Command: list-acl

The `list-acl` command provides an easy way of knowing the authorization configuration of a running PAP.

Typing:

```
pap-admin list-acl
```

prints out the Access Control Entries (ACEs) comprising the ACL currently defined for the running PAP.

Example:

```
~# pap-admin list-acl

/voms-ws/Role=PAP-Admin :
   POLICY_READ_LOCAL|POLICY_READ_REMOTE|POLICY_WRITE|CONFIGURATION_READ|CONFIGURATION_WRITE

"/C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Andrea Ceccanti" :
   POLICY_READ_LOCAL|POLICY_READ_REMOTE|POLICY_WRITE|CONFIGURATION_READ|CONFIGURATION_WRITE

ANYONE :
   CONFIGURATION_READ|CONFIGURATION_WRITE
```

Required permissions : `CONFIGURATION_READ`.

Command: add-ace

The `add-ace` command allows to add (or change) an ACE to the PAP ACL. Note that if an ACE entry already exists on the server for the principal specified in the command, the permissions in such ACE are **replaced** by the ones specified in the command.

Usage:

```
pap-admin add-ace <principal> <permissions>
```

where:

- `principal` can be either an X509 DN or a VOMS FQAN. `ANYONE` can be used to assign permissions to any authenticated user.
- `permissions` is a | separated list of PAP permissions that will be assigned to `principal`. The `ALL` shortcut can be used to assign all permission. Valid permissions values are defined here.

Example:

```
pap add-ace '/atlas/Role=VO-Admin' 'ALL'
```

Required permissions: `CONFIGURATION_READ|CONFIGURATION_WRITE`

Command: set-polling-interval

Command: remove-ace

The `remove-ace` command removes an ACE from the PAP ACL.

Usage:

```
pap-admin remove-ace <principal>
```

where:

- `principal` can be either an X509 DN or a VOMS FQAN. `ANYONE` can be used to remove permissions assigned to any authenticated user.

Example:

```
pap remove-ace '/atlas/Role=VO-Admin'
```

Required permissions: `CONFIGURATION_READ|CONFIGURATION_WRITE`

The Simplified Policy Language

As already explained here, Argus policies contain collections of rules that state which actions can be performed on which resources by which users. XACML, the language used internally by Argus to define policies, provides great expressiveness and flexibility but it's very hard to read and author for uman beings. For this reason, Argus provides a Simplified Policy Language (SPL) to hide the complexity of XACML while providing much of its flexibility.

As an example, the following policy denies access to all the resources (under Argus control) to the members of the ATLAS VO:

```
resource ".*" {
   action ".*" {
      rule deny { vo = "atlas" }
   }
}
```

The SPL syntax

```
resource <value> {

   action <value> {

      rule <permit|deny> {

         <attributeId>=<attributeValue>
         ...
      }
      ...
   }
   ...
}
...
```

The SPL defines three stanza types: `resource`, `action` and `rule`. It's possible to define multple resouce stanzas that can contain multiple action stanzas that can contain multiple rules stanzas.

The `resource` stanza is used to target a resource (or set of resources, if wildcards are used) under the control of Argus authorization.

The `action` stanza (always defined in the context of an enclosing `resource`) is used to target an action (or set of actions, if wildcards are used) that has to be authorized by Argus on the resource identified by the enclosing `resource` stanza.

The `rule` stanza defines who is authorized (in case of a `permit` rule) or not authorized (in case of a `deny` rule) to perform the action on the resource identified by the enclosing action and resource stanzas.

Identifying actions and resources

Actions and resources are identified by unique identifiers that are assigned to them. This identifiers are usually URIs, but any string that is unique in your deployment may work.

You can also use wildcards in your SPL policies to target group of resources or actions, like in the following example:

```
resource "http://cnaf.infn.it/cream-ce-01" {
```

```
   action ".*" {

      rule permit { vo = "cms" }

   }
}
```

This policy authorizes users from the CMS vo to perform any action on the resource `http://cnaf.infn.it/cream-ce-01`.

Identifying subjects

In Argus policies, the users (or software agents) that need to be authorized to execute an action on a specific resource are identified using a set of attributes, like:

- the subject of the user's X509 certificate;
- the CA that issued the user's x509 certificate;
- the VO the user belongs to;
- whether the use has a specific FQAN in its bag of VOMS attributes;
- whether the user has a specific FQAN as his primary FQAN.

The table below specifies the supported attributes for Argus 1.1:

Attribute name	Description	Example
`subject`	The user's X509 certificate subject in rfc2253 or openssl format	`subject ="CN=Andrea Ceccanti,L=CNAF,OU=Personal Certificate,O=INFN,C`
`subject-issuer`	The subject (in rfc2254 or openssl format) of the CA that issued the user's x509 certificate	`subject-issuer = "CN=INFN CA,O=INFN,C=IT"`
`vo`	The name of the VO the user belongs to	`vo = "atlas"`
`fqan`	The fqan present in the user's bag of VOMS attributes	`fqan="/dteam/Role=VO-Admin"`
`pfqan`	The user primary fqan	`pfqan="/atlas/Role=pilot"`

The contents of the `rule` stanza

As already pointed out, the `rule` stanza defines who is authorized to perform a specific action on a specific resource. A subject can be identified using the attributes defined in the previous section.

```
resource "http://cnaf.infn.it/cream-ce-01" {

   action "submit-pilot-job" {

      rule permit { pfqan="/atlas/Role=pilot" }

   }
}
```

In the above policy, only subjects that have the `/atlas/Role=pilot` fqan as their primary fqan are authorized (since the rule is `permit` rule) to perform the action `submit-pilot-job` on the resource `http://cnaf.infn.it/cream-ce-01`. To prevent users from LHCB VO the execution of the same action, one would write the following policy:

```
resource "http://cnaf.infn.it/cream-ce-01" {

   action "submit-pilot-job" {

      rule deny { vo = "lhcb" }

   }
}
```

Multiple attributes inside the `rule` stanza

It is possibile to define multiple attributes inside a `rule` stanza. All the attributes defined in the rule stanza need to match with the subject attributes present in the authorization request for the rule to be applied. This can be explained more clearly using an example:

```
resource "http://cnaf.infn.it/cream-ce-01" {

   action "submit-job" {

      rule permit {
         vo = "cms"
         subject-issuer = "CN=INFN CA,O=INFN,C=IT"
      }
   }
}
```

The meaning of the above policy is that only members from the VO CMS that have a certificate signed by the `CN=INFN CA,O=INFN,C=IT` CA will be authorized to perform the action `submit-job` on resource `http://cnaf.infn.it/cream-ce-01`. CMS members with certificates signed by the CERN CA, for instance, will not be authorized.

Since all the attributes defined in a rule must be "matched" in the request for the rule to be applied, one can think about multiple attributes inside a rule stanza as conditions that are ANDed to select who will be authorized to perform the action the rule is about.

How policies are evaluated

The first applicable policy (and only that one) that matches the authorization request is the one that is applied by Argus. This means that **order matters**. An example will help in understanding this concept.

Suppose we want to grant access to our CE to all members of VO CMS but not those that have `/cms/Role=pilot` as their primary FQAN. We would write a policy like this:

```
resource "http://cnaf.infn.it/cream-ce-01" {

   action ".*" {

      rule deny{ pfqan = "/cms/Role=pilot"}
      rule permit { vo = "cms" }

   }
}
```

Since the deny rule precedes the permit rule in the above policy, we are able to deny access only to CMS users with the pilot role, but grant access to other members of CMS. This is due to the fact that the first deny rule will not match to CMS users that do not have the pilot role, so the following permit rule will be applied. On the contrary, if we reversed the order of the two rules like in the following policy:

```
resource "http://cnaf.infn.it/cream-ce-01" {

   action ".*" {

      rule permit { vo = "cms" }
      rule deny{ pfqan = "/cms/Role=pilot" }
   }
}
```

the deny rule would be useless, since the permit rule that precedes it would always match any CMS member.

The `obligation` stanza

Starting with Argus version 1.1, the SPL supports `obligation` stanzas. The syntax of the obligation stanza is as follows:

```
obligation "obligationId" {
   [attributeId = attributeValue]*
}
```

Oligation stanzas can be placed either in the resource or action context and are used to define a set operations that must be performed by the Argus PEP in conjuction with an authorization decision. An obligation stanza can define 0..N attribute definitions, that are passed as parameters to the PEP for the fulfillment of the obligation.

An example of policy with an obligation is the following:

```
resource "http://cnaf.infn.it/wn"{

   obligation "http://glite.org/xacml/obligation/local-environment-map" {}

   action "http://glite.org/xacml/action/execute"{
      rule permit { vo = "dteam" }
   }
}
```

The Argus PEP currently supports only the `map-to-local-enviroment` obligation.

The `map-to-local-environment` obligation

The `map-to-local-environment` obligation, identified by the following id:

```
http://glite.org/xacml/obligation/local-environment-map
```

is used within a policy to signify that a mapping to a local posix account will be produced by the Argus server as a result of a permit policy.

The use of this obligation is **mandatory** for the policies that authorize the execution and mapping of pilot jobs on the worker node.

Examples

Ban policies

Ban policies are used to deny a subject on all possible resources. For this reason ban policies need to be placed at the top and defined for any action on all the resources.

```
resource ".*" {
    action ".*" {
        rule deny { subject = "CN=Alberto Forti,L=CNAF,OU=Personal Certificate,O=INFN,C=IT" }
        rule deny { fqan = /dteam/test }
    }
}
```

Glexec on the WN policies

Policy that authorize execution and mapping of pilot jobs on the WN need to specify the `map-to-local-environment` obligation to produce a mapping that gLexec can use to do the user switch. An example of such policy is the following:

```
resource "http://cnaf.infn.it/wn"{

   obligation "http://glite.org/xacml/obligation/local-environment-map" {}

   action "http://glite.org/xacml/action/execute"{
      rule permit { vo = "dteam" }
      rule permit { pfqan = "/atlas/Role=pilot" }
      rule permit { pfqan = "/ops/Role=pilot" }
   }
}
```

The above policy authorizes the execution of jobs on the WN by:

- people from the dteam VO,
- people that have `/atlas/Role=pilot` as the primary fqan
- people that have `/ops/Role=pilot` as the primary fqan

Argus Policy Decision Point (PDP): Configuration

Configuration File Syntax

The PDP is configured through the use of the `pdp.ini` file. This file is a standard INI file with three defined sections. The `SERVICE` section contains properties related the PDP service as a whole and how it listens for incoming requests. The `POLICY` section contains properties for the retrieval of policies from the Policy Administration Point (PAP). The final section, `SECURITY`, contains properties that related to various security aspects of the service, the services private key and certificate, for example.

Basic Configuration Options

SERVICE section

Property	Description	Required?	Default Value
entityID	This is a unique identifier for the PDP. It must be a URI (URL or URN) and the same entity ID should be used for all PDP instances that make up a single logical PDP. If a URL is used it need not resolve to any specific webpage.	Y	None. Recommended value is a URL corresponding to the logical PDP service (e.g. `http://pdp.example.org`).
hostname	This is the hostname or IP address to which the service will bind.	Y	None.
port	This is the port to which the service will bind.	N	8152
adminHost	The hostname upon which the service will listen for admin commands.	N	127.0.0.1
adminPort	This is the port upon which the service will listen for admin command.	N	8153
adminPassword	This is the password required to accompany admin commands. If unspecified than no password is required to run admin commands.	N	None

POLICY section

Property	Description	Required?	Default Value
paps	A space separated list of PAP endpoint URLs. Endpoints will be tried in turn until one returns a successful response. This provides limited failover support. If more intelligent failover is necessary or load balancing is required, a dedicated load-balancer/failover appliance should be used.	Y	None
retentionInterval	The number of minutes the PDP will retain (cache) a policy retrieved from the PAP. After this time is passed the PDP will again call out to the PAP and retrieve the policy.	N	240 (4 hours)

SECURITY section

Property	Description	Required?	Default Value
servicePrivateKey	An absolute path to the file containing the unencrypted, PEM-encoded, private key used by this service. All PDPs instances within a single logical PDP should use the same key.	Y	None.
serviceCertificate	An absolute path to the file containing the unencrypted, PEM-encoded, certifcate used by this service. All PDPs instances within a single logical PDP should use the same key.	Y	None.
trustInfoDir	An absolute path to the directory that contains standard X.509 trust information, such as the IGTF Trust Anchor Distribution.	Y	None
enableSSL	Indicates whether the service port should use SSL/TLS or not.	N	false
requireClientCertAuthentication	Indicates whether the client must use a valid client certificate to authenticate to the PDP	N	false

Example pdp.ini files

The following example file contain the bare minimum required for a valid PDP configuration file.

```
[SERVER]
entityID = http://argus.example.org/pdp
hostname = argus.example.org

[POLICY]
paps = https://argus.example.org:8150/pap/services/ProvisioningService

[SECURITY]
servicePrivateKey = /etc/grid-security/hostkey.pem
serviceCertificate = /etc/grid-security/hostcert.pem
trustInfoDir = /etc/grid-security/certificates
# HTTPS enabled
enableSSL = true
```

Advanced Configuration Options

The following advanced options are available but are unlikely to ever be used by deployers. They are mostly for performing very fine-grained tuning of request/response handling parameters. Incorrectly configuring these can have a very negative impact on performance so deployers should not change these unless they are very sure they understand what the impact will be.

SERVICE section

Property	Description	Required?	Default Value
maximumRequests	The maximum number of requests that will be processed simultaneously. Additional requests will be queued.	N	200
requestQueueSize	The maximum number of requests that will be queued up when all the processing threads are busy. Incoming requests received when all processing threads are busy and the queue is full will receive an HTTP 503 error.	N	500

connectionTimout	This is the length of time, in seconds, the service will wait for the client to send information before it considers the request timed out.	N	30 seconds
receiveBufferSize	This is the size, in bytes, that will be allocated to the HTTP request buffer.	N	16384 (16kb)
sendBufferSize	This is the size, in bytes, that will be allocated to the HTTP response buffer.	N	16384 (16kb)

POLICY section

Property	Description	Required?	Default Value
policySetId	The ID of the policy to fetch from the PAP.	N	-1
connectionTimout	This is the length of time, in seconds, the PAP client will wait for the PAP to send information before it considers the request timed out.	N	30 seconds
receiveBufferSize	This is the size, in bytes, that will be allocated to the PAP client send buffer.	N	16384 (16kb)
sendBufferSize	This is the size, in bytes, that will be allocated to the PAP client request buffer.	N	16384 (16kb)

SECURITY section

Property	Description	Required?	Default Value
trustInfoRefresh	The frequency, in minutes, that the trust material specified by `trustInfoDir` will be checked for updates.	N	60 (1 hour)
messageValidityPeriod	The number of seconds, from the time a message is issued, until it is considered expired.	N	300s (5 minutes)

| clockSkew | The allowance, in seconds, used when computing validity periods. | N | 30s |

Argus PEP Server: Configuration

Configuration File

The PEP is configured through the user of the `pepd.ini` file. This file is a standard INI file with different defined sections. The `SERVICE` section contains properties related the PEP Server service as a whole and how it listens for incoming requests. The `PDP` section that control how callouts to the PDP are made. The final section, `SECURITY`, contains properties that related to various security aspects of the service, the services private key and certificate, for example.

PIP Configuration

Each Policy Information Point (PIP) is configured in its own INI section, referenced in the `SERVICE` section under the `pips` property.

The configuration properties needed by each PIP is specific to that given PIP. Some will not require any properties beyond the standard set while other will need addition information.

- See the Policy Information Point documentation for more information:
  - Request Validator PIP
  - OpenSSL Subject Converter PIP
  - gLite Grid Authorization Profile PIP
  - Common XACML Authorization Profile PIP

Obligation Handler Configuration

Each Obligation Handler (OH) is configured in its own INI section, referenced in the `SERVICE` section under the `obligationHandlers` property.

The configuration properties needed by each Obligation Handler is specific to that given OH. Some will not require any properties beyond the standard set while other will need addition information.

- See the Obligation Handler documentation for more information:
  - Gridmap Account Mapping Obligation Handler

Basic Configuration Options

SERVICE section

Property	Description	Required?	Default Value
entityID	This is a unique identifier for the PEP. It must be a URI (URL or URN) and the same entity ID should be used for all PEP instances that make up a single logical PEP. If a URL is used it need not resolve to any specific webpage.	Y	None. Recommended value is a URL corresponding to the logical PEP service (e.g. `http://pep.example.org`).
hostname	This is the hostname or IP address to which the service will bind.	Y	None
port	This is the port to which the	N	8154

	service will bind.		
adminHost	The hostname upon which the service will listen for admin commands.	N	127.0.0.1
adminPort	This is the port upon which the service will listen for admin commands. This port is only available on the localhost (127.0.0.1).	N	8155
adminPassword	This is the password required to accompany admin commands. If unspecified than no password is required to run admin commands.	N	None
pips	This is a space separated list of the INI section names that configure policy information points (PIP) that the PEP will invoke upon the arrival of every request. PIPs are executed in the order listed by this property. See the policy information point documentation for more information.	N	None
obligationHandlers	This is a space separated list of the INI section names that configure obligations handlers that the PEP will use to fulfill obligation requirements sent back by the PDP. See the obligation handler documentation for more information.	N	None

PDP section

Property	Description	Required?	Default Value
pdps	A space separated list of PDP endpoint URLs. Endpoints will be tried in turn until one returns a successful response. This provides limited failover support. If more intelligent failover is necessary or load balancing is required, a dedicated load-balancer/failover appliance should be used.	Y	None
maximumCachedResponses	The maximum number of responses from any PDP that will be cached. Setting this value to 0 (zero) will disable caching. The maximum amount of time a single response is cached is controlled by the `cachedResponseTTL` property described below.	N	500

SECURITY section

Property	Description	Required?	Default Value
servicePrivateKey	An absolute path to the file containing the unencrypted, PEM-encoded, private key used by this service.	Yes, if requests from the PEP client should be	None.

		done over HTTPS.	
serviceCertificate	An absolute path to the file containing the unencrypted, PEM-encoded, certificate used by this service.	Yes, if requests from the PEP client should be done over HTTPS.	None.
trustInfoDir	An absolute path to the directory that contains standard X.509 trust information, such as the IGTF Trust Anchor Distribution.	Required when connecting to PDPs over HTTPS.	None
enableSSL	Enable HTTPS on the service port (SSL/TLS). The `serviceCertificate`, `servicePrivateKey`, and `trustInfoDir` properties must also be defined in order to use this setting.	N	false
requireClientCertAuthentication	The client must have a valid X.509 client certificate to authenticate to the PEP Server	N	true

Example pepd.ini files

The following example file contain the bare minimum required for a valid PEP configuration file.

```
[SERVICE]
entityId = https://argus.example.org/pep
hostname = argus.example.org

[PDP]
pdps = https://argus.example.org:8152/authz

[SECURITY]
```

The following example file contains the bare minimum required for a valid PEP configuration plus the configuration of a couple PIPs. Note how each element in the list `pips` list of the `SERVER` section matches the name section configuring the PIP. Also note that the `REQVALIDATOR_PIP` takes a few additional configuration parameters.

```
[SERVICE]
entityID = http://argus.example.org/pep
hostname = argus.example.org
pips = REQVALIDATOR_PIP

[PDP]
pdps = https://argus.example.org:8152/authz https://pdp2.example.org:8152/authz

[SECURITY]
servicePrivateKey = /etc/grid-security/hostkey.pem
serviceCertificate = /etc/grid-security/hostcert.pem
trustInfoDir = /etc/grid-security/certificates
enableSSL = true
requireClientCertAuthentication = true

[REQVALIDATOR_PIP]
parserClass = org.glite.authz.pep.pip.provider.RequestValidatorPIPIniConfigurationParser
validateRequestSubjects = true
validateRequestResources = true
validateRequestAction = true
validateRequestEnvironment = false
```

SECURITY section

Advanced Configuration Options

The following advanced options are available but are unlikely to ever be used by deployers. They are mostly for performing very fine-grained tuning of request/response handling parameters. Incorrectly configuring these can have a very negative impact on performance so deployers should not change these unless they are very sure they understand what the impact will be.

SERVICE section

Property	Description	Required?	Default Value
maximumRequests	The maximum number of requests that will be processed simultaneously. Additional requests will be queued.	N	200
requestQueueSize	The maximum number of requests that will be queued up when all the processing threads are busy. Incoming requests received when all processing threads are busy and the queue is full will receive an HTTP 503 error.	N	500
connectionTimout	This is the length of time, in seconds, the service will wait for the client to send information before it considers the request timed out.	N	30 seconds
receiveBufferSize	This is the size, in bytes, that will be allocated to the HTTP request buffer.	N	16384 (16kb)
sendBufferSize	This is the size, in bytes, that will be allocated to the HTTP response buffer.	N	16384 (16kb)

PDP section

Property	Description	Required?	Default Value
maximumRequests	The maximum number of simultaneous requests that will be made to the PDP. Additional requests will wait until a free request slot becomes available.	N	200
cachedResponseTTL	The length of time, in seconds, for which a response will be cached.	N	600 seconds (10 minutes)
connectionTimout	This is the length of time, in seconds, the PAP client will wait for the PAP to send information before it considers the request timed out.	N	30 seconds
receiveBufferSize	This is the size, in bytes, that will be allocated to the PDP client send buffer.	N	16384 (16kb)
sendBufferSize	This is the size, in bytes, that will be allocated to the PDP client request buffer.	N	16384 (16kb)

SECURITY section

Property	Description	Required?	Default Value
trustInfoRefresh	The frequency, in minutes, that the trust material specified by `trustInfoDir` will be checked for updates.	N	60 (1 hour)

Argus PEP Server Policy Information Points (PIP)

Policy Information Points (PIPs) are plugins to the authorization service that help populate and/or complete an authorization request. PIPs may rely on information already within the request or they may simply be able to self generate the data that they will add.

Request Validator PIP

NOTE: This PIP is new in Argus 1.3 (EMI).

The Request Validator PIP validates the incoming authorization request. It checks that the incoming authorization request contains at least one subject, one resource and one action, and that the attributes within them have at least one value, which is not null or an empty string.

Configuration

1. Create a new INI section for you PIP (you may choose any valid INI section name. e.g. REQVALIDATOR_PIP)
2. Into the PIP INI section add the `parserClass` property with the value `org.glite.authz.pep.pip.provider.RequestValidatorPIPIniConfigurationParser`
3. Configure how the authorization request must be validated

PIP Configuration Properties

Property	Description	Required?	Default Value
validateRequestSubjects	Require at least one subject with non-empty attribute values	N	`true`
validateRequestResources	Require at least one resource with non-empty attribute values	N	`true`
validateRequestAction	Require one action with non-empty attribute values	N	`true`
validateRequestEnvironment	Require one environment with non-empty attribute values	N	`false`

Example Configuration

The following example shows a PEP server configuration with the request validator PIP enabled. The PIP validates the incoming authorization request and checks that it contains at least one subject, at least one resource and one action, all with non-empty or null attributes values.

```
[SERVICE]
entityId = https://example.org/pep
hostname = example.org
pips = REQVALIDATOR_PIP

[PDP]
pdps = http://localhost:8152/authz

[SECURITY]
trustInfoDir = /etc/grid-security/certificates

[REQVALIDATOR_PIP]
parserClass = org.glite.authz.pep.pip.provider.RequestValidatorPIPIniConfigurationParser
validateRequestSubjects = true
validateRequestResources = true
validateRequestAction = true
```

```
validateRequestEnvironment = false
```

OpenSSL Subject Converter PIP

NOTE: This PIP is new in Argus 1.3 (EMI).

The OpenSSL Subject Converter PIP transforms on-the-fly an incoming request subject attribute *subject-id* and/or attribute *subject-issuer* value from the old, unsupported and wrong OpenSSL oneline format (e.g. "/C=CH/O=example.org/CN=John Doe") into a correct RFC2253 format value (e.g. "CN=John Doe,O=example.org,C=CH") with the correct datatype.

Configuration

1. Create a new INI section for you PIP (you may choose any valid INI section name. e.g. OPENSSLSUBJECT_PIP)
2. Into the PIP INI section add the `parserClass` property with the value `org.glite.authz.pep.pip.provider.OpenSSLSubjectPIPIniConfigurationParser`
3. Configure which subject attribute ID and datatype values must be transformed from the OpenSSL format into the RFC2253 format.

PIP Configuration Properties

Property	Description	Required?	Default Value
openssslSubjectAttributeIDs	The space separated list of subject attribute IDs containing an OpenSSL value to convert	No	`urn:oasis:names:tc:xacml:1.0:subject:s` `http://glite.org/xacml/attribute/subje`
openssslSubjectAttributeDatatypes	The space separated list of subject attribute datatypes containing an OpenSSL value to convert	No	`http://www.w3.org/2001/XMLSchema#strin`

Example Configuration

The following example shows a PEP server configuration with the OpenSSL Subject Converter PIP enabled, and transforming both the subject attribute IDs `urn:oasis:names:tc:xacml:1.0:subject:subject-id` and `http://glite.org/xacml/attribute/subject-issuer`, with the datatype `http://www.w3.org/2001/XMLSchema#string` values from the OpenSSL oneline format into the RFC2253 format.

```
[SERVICE]
entityId = https://example.org/pep
hostname = example.org
pips = OPENSSLSUBJECT_PIP
```

Example Configuration

```
[PDP]
pdps = http://localhost:8152/authz

[SECURITY]
trustInfoDir = /etc/grid-security/certificates

[OPENSSLSUBJECT_PIP]
parserClass = org.glite.authz.pep.pip.provider.OpenSSLSubjectPIPIniConfigurationParser
opensslSubjectAttributeIDs = http://glite.org/xacml/attribute/subject-issuer urn:oasis:names:tc:x
opensslSubjectAttributeDatatypes = http://www.w3.org/2001/XMLSchema#string
```

gLite Grid Authorization Profile PIP

NOTE: This is the default profile supported starting from Argus 1.2.

This PIP allows the PEP client to send only the end-user certificate or proxy as lone Subject *Key-Info* attribute. It will then parse the certificate, extract all the information from the certificate required by the gLite Grid XACML Authorization Profiles, and populate the request with attributes found in the certificate/proxy.

This PIP implements the XACML Grid Worker Node Authorization Profile (v.1.0) and the XACML Grid Computing Element Authorization Profile (v.1.0) specifications.

Configuration

1. Create a new INI section for you PIP (you may choose any valid INI section name. e.g. GLITEXACMLPROFILE_PIP)
2. Into the PIP INI section add the `parserClass` property with the value `org.glite.authz.pep.pip.provider.GLiteAuthorizationProfilePIPIniConfigurat`
3. To enable VOMS attribute certificate support add the `vomsInfoDir` property with a value corresponding to the absolute path of the VOMS `vomsdir`, traditionally `/etc/grid-security/vomsdir`.
4. If, in the `SECURITY` section, the `trustInfoDir` property is not already set, add it with a value of the absolute filesystem path of your IGTF trust bundle.
5. Configure which profile IDs are to be accepted.

PIP Configuration Properties

Property	Description	Required?	Default Value
acceptedProfileIDs	The space separated list of accepted authorization profile IDs	No	None.
vomsInfoDir	The absolute path to the VOMS `vomsdir` directory.	Y	None.
vomsInfoRefresh	The refresh interval time in minutes of the `vomsInfoDir` directory.	No	60
requireCertificate	The request Subject attribute key-info MUST be present in the incoming request.	No	`true`
requireProxy	The request Subject attribute key-info MUST to be a proxy (PEM encoded proxy chain).	No	`false`

NOTE: If the `acceptedProfileIDs` is not defined, then all profile IDs present in the request environment *profile-id* attribute are accepted.

Required Request Attributes

This PIP requires that the request environment contains a *profile-id* attribute with the profile identifier, and that the request subject contains the certificate, and its chain, that were used to authenticate to the service, in

Example Configuration

the *key-info* attribute:

- The Profile Identifier
  - **type:** Environment
  - **id:** http://glite.org/xacml/attribute/profile-id
  - **data type:** http://www.w3.org/2001/XMLSchema#anyURI
  - **multiple values allowed:** no
  - **description:** The profile ID implemented by the incoming request.

- The Certificate or Proxy Certificate (with chain)
  - **type:** Subject
  - **id:** urn:oasis:names:tc:xacml:1.0:subject:key-info
  - **data type:** http://www.w3.org/2001/XMLSchema#string
  - **multiple values allowed:** no
  - **description:** The PEM encoded certificate chain. No certificate order is assumed however all certificates must be version 3 certificates. Zero or one VOMS attribute certificate may also be included.

Populated Effective Request Attributes

The PIP will process the request subject *key-info* attribute and populate the following attributes:

- The Subject Identifier
  - **type:** Subject
  - **id:** urn:oasis:names:tc:xacml:1.0:subject:subject-id
  - **data type:** urn:oasis:names:tc:xacml:1.0:data-type:x500Name
  - **multiple values allowed:** no
  - **description:** This is the Subject DN as given in the end-entity certificate. It is in RFC2253 format.

- The End-entity Certificate Issuer
  - **type:** Subject
  - **id:** http://glite.org/xacml/attribute/subject-issuer
  - **data type:** urn:oasis:names:tc:xacml:1.0:data-type:x500Name
  - **multiple values allowed:** yes
  - **description:** This is the Subject DN of the root CA and all subordinate CAs that signed within the end-entity certificate chain. It is in RFC2253 format.

If VOMS support is enabled and a VOMS certificate is included within a user's proxy certificate, the following attributes will be populated within the request:

- The VO Name
  - **type:** Subject
  - **id:** http://glite.org/xacml/attribute/virtual-organization
  - **data type:** http://www.w3.org/2001/XMLSchema#string
  - **multiple values allowed:** yes
  - **description:** The names of the VOs to which the user is a member. Currently there is only ever one value.

- The VOMS Primary FQAN
  - **type:** Subject
  - **id:** http://glite.org/xacml/attribute/fqan/primary
  - **data type:** http://glite.org/xacml/datatype/fqan
  - **issuer:** DN of the attribute certificate issuer

Required Request Attributes

- **multiple values allowed:** no
- **description:** The primary Fully Qualified Attribute Name (FQAN) for the subject

- The VOMS FQANs
  - **type:** Subject
  - **id:** http://glite.org/xacml/attribute/fqan
  - **data type:** http://glite.org/xacml/datatype/fqan
  - **multiple values allowed:** yes
  - **description:** All the Fully Qualified Attribute Name (FQAN)s for the subject

Example Configuration

The following example shows a PEP Server configuration with the Grid authorization profile PIP enabled, and accepting both the `http://glite.org/xacml/profile/grid-ce/1.0` and the `http://glite.org/xacml/profile/grid-wn/1.0` XACML Grid authorization profiles.

```
[SERVICE]
entityId = https://example.org/pep
hostname = example.org
pips = GLITEXACMLPROFILE_PIP

[PDP]
pdps = http://localhost:8152/authz

[SECURITY]
trustInfoDir = /etc/grid-security/certificates

[GLITEXACMLPROFILE_PIP]
parserClass = org.glite.authz.pep.pip.provider.GLiteAuthorizationProfilePIPIniConfigurationParser
vomsInfoDir = /etc/grid-security/vomsdir
acceptedProfileIDs = http://glite.org/xacml/profile/grid-ce/1.0 http://glite.org/xacml/profile/gr
```

Common XACML Authorization Profile PIP

NOTE: This profile is supported since Argus 1.6 (EMI-3).

This PIP allows the PEP client to send only the end-user certificate or proxy as lone Subject *Key-Info* attribute. It will then parse the certificate, extract all the information from the certificate required by the Common XACML Authorization Profile, and populate the request with attributes found in the certificate/proxy.

This PIP implements the Common XACML Authorization Profile (1.1.1) specifications.

Configuration

1. Create a new INI section for you PIP (you may choose any valid INI section name. e.g. COMMONXACMLPROFILE_PIP)
2. Into the PIP INI section add the `parserClass` property with the value `org.glite.authz.pep.pip.provider.CommonXACMLAuthorizationProfilePIPIniConf`
3. To enable VOMS attribute certificate support add the `vomsInfoDir` property with a value corresponding to the absolute path of the VOMS `vomsdir`, traditionally `/etc/grid-security/vomsdir`.
4. If, in the `SECURITY` section, the `trustInfoDir` property is not already set, add it with a value of the absolute filesystem path of your IGTF trust bundle.
5. Configure which profile IDs are to be accepted, normally `http://dci-sec.org/xacml/profile/common-authz/1.1`

Populated Effective Request Attributes

PIP Configuration Properties

Property	Description	Required?	Default Value
acceptedProfileIDs	The space separated list of accepted authorization profile IDs	No	None.
vomsInfoDir	The absolute path to the VOMS `vomsdir` directory.	YES	None.
vomsInfoRefresh	The refresh interval time in minutes of the `vomsInfoDir` directory.	No	60
requireCertificate	The request Subject attribute key-info MUST be present in the incoming request.	No	`false`
requireProxy	The request Subject attribute key-info MUST to be a proxy (PEM encoded proxy chain).	No	`false`

NOTE: If the `acceptedProfileIDs` is not defined, then all profile IDs present in the request environment *profile-id* attribute are accepted.

Required Request Attributes

This PIP requires that the request environment contains a *profile-id* attribute with the profile identifier, and that the request subject contains the certificate, and its chain, that were used to authenticate to the service, in the *key-info* attribute:

- The Profile Identifier Attribute
  - **type:** Environment
  - **id:** http://dci-sec.org/xacml/attribute/profile-id
  - **data type:** http://www.w3.org/2001/XMLSchema#anyURI
  - **multiple values allowed:** no
  - **description:** The profile ID implemented by the incoming request, typically `http://dci-sec.org/xacml/profile/common-authz/1.1`

- The Subject Key-Info (certificate or proxy, with chain) Attribute
  - **type:** Subject
  - **id:** urn:oasis:names:tc:xacml:1.0:subject:key-info
  - **data type:** http://www.w3.org/2001/XMLSchema#base64Binary
  - **multiple values allowed:** yes
  - **description:** The multiple values are the base64 encoded DER blocks of the certicate/proxy chain.

Populated Effective Request Attributes

The PIP will process the request subject *key-info* attribute and populate the following attributes:

- The Subject Identifier Attribute
  - **type:** Subject
  - **id:** urn:oasis:names:tc:xacml:1.0:subject:subject-id
  - **data type:** urn:oasis:names:tc:xacml:1.0:data-type:x500Name
  - **multiple values allowed:** no
  - **description:** X.509 distinguished name of the end-entity certificate. The value is in RFC2253 format, e.g. "CN=John Doe,DC=example,DC=org"

- The Subject Issuer Attribute
  - **type:** Subject
  - **id:** http://dci-sec.org/xacml/attribute/subject-issuer
  - **data type:** urn:oasis:names:tc:xacml:1.0:data-type:x500Name
  - **multiple values allowed:** yes

- **description:** X.509 distinguished name of the authority(ies) which issued the end-entity certificate. The values are in RFC2253 format.

If VOMS support is enabled and a VOMS certificate is included within a user's proxy certificate, the following attributes will be populated within the request:

- The Virtual Organization (VO) Attribute
  - **type:** Subject
  - **id:** http://dci-sec.org/xacml/attribute/virtual-organization
  - **data type:** http://www.w3.org/2001/XMLSchema#string
  - **multiple values allowed:** yes
  - **description:** The names of the VOs to which the user is a member. Currently there is only ever one value.

- The Primary Group and Group Attributes
  - **type:** Subject
  - **id:** http://dci-sec.org/xacml/attribute/group/primary and http://dci-sec.org/xacml/attribute/group
  - **data type:** http://www.w3.org/2001/XMLSchema#string
  - **multiple values allowed:** no (primary group), yes (groups)
  - **description:** The primary group name, and the list of all group names

- The Primary Role and Role Attributes
  - **type:** Subject
  - **id:** http://dci-sec.org/xacml/attribute/role/primary and http://dci-sec.org/xacml/attribute/role
  - **data type:** http://www.w3.org/2001/XMLSchema#string
  - **issuer:** The group name to which this role belong.
  - **multiple values allowed:** no (primary role), yes (roles)
  - **description:** The primary role, and roles list assigned to the subject.

Example Configuration

The following example shows a PEP Server configuration with the Common XACML authorization profile PIP enabled, and accepting the `http://dci-sec.org/xacml/profile/common-authz/1.1` EMI Common XACML Authorization profile.

```
[SERVICE]
entityId = https://argus.example.org/pep
hostname = argus.example.org

pips = COMMONXACMLPROFILE_PIP

[PDP]
pdps = https://argus.example.org:8152/authz

[SECURITY]
trustInfoDir = /etc/grid-security/certificates

[COMMONXACMLPROFILE_PIP]
parserClass = org.glite.authz.pep.pip.provider.CommonXACMLAuthorizationProfilePIPIniConfiguration
vomsInfoDir = /etc/grid-security/vomsdir
acceptedProfileIDs = http://dci-sec.org/xacml/profile/common-authz/1.1
```

Populated Effective Request Attributes

Argus PEP Server Obligation Handlers

Obligation handlers are plugins used by the authorization service in order to adjust the environment, under which an action will run, to a state that meets a particular set of obligations. Obligation are things like "write output to directory X" or "perform all work as user 1".

Grid Map POSIX Account Mapping Obligation Handler

NOTE: The Grid Map Account Mapping Obligation Handler only works with the gLite Grid Authorization Profile PIP, or with clients implementing the XACML Grid Worker Node Authorization Profile (v.1.0) or the XACML Grid Computing Element Authorization Profile (v.1.0) specifications.

This obligation handler maps a subject ID, given as a DN, and set of FQANs, for example those provided by the gLite Grid Authorization Profile PIP, in to a POSIX account (login name, primary group name and secondary group names).

This mapping is controlled by two grid map files, one that provides the mapping of the subject to an account indicator (login name or pool account indicatior) and one that maps the subject to a set of group names (one primary and any number of secondary).

Configuration

1. Create the account and group mapping files appropriate for your environment
2. Create a new INI section for your handler (you may choose any valid INI section name)
3. To handler INI section add the `parserClass` property with the value `org.glite.authz.pep.obligation.dfpmap.DFPMObligationHandlerConfigurationPa`
4. Define at least the required (`accountMapFile`, `groupMapFile`, `gridMapDir`) and any optional configuration properties for this handler

OH Configuration Properties

Property	Description	Required?	Default
`accountMapFile`	The absolute path to the map file used to map a subject to a POSIX login name.	Y	None.
`groupMapFile`	The absolute path to the map file used to map a subject to a set of POSIX groups.	Y	None.
`gridMapDir`	The absolute path to the grid map directory.	Y	None.
`handledObligationId`	The identifier of	N	`http://glite.org/xacml/obliga`

	the obligation handled by the handler. The obligation handler is triggered only if the obligation ID match this value.		
`preferDNForLoginName`	Indicates whether to prefer a DN based mapping for the login name mapping over a primary FQAN login name mapping.	N	`true`
`preferDNForPrimaryGroupName`	Indicates whether to prefer a DN based mapping for the primary group name mapping over a primary FQAN group name mapping	N	`true`
`noPrimaryGroupNameIsError`	Indicates that the failure to find a primary group mapping in the group map file cause the obligation handler to fail..	N	`false`
`refreshPeriod`	The period, in minutes, between when the	N	15

	map files are checked, and if they have been changed, reread.		
`requireSubjectKeyInfo`	The obligation handler will only be applied if the request subject contains a *key-info* attribute (PEM encoded certificate)	N	`true`
`useSecondaryGroupNamesForMapping`	The obligation handler will create lease file names containing the secondary groups of the user	N	`true`

NOTE: the default `preferDNForLoginName` property value was *false* for Argus 1.0 and Argus 1.1. From Argus 1.2 the default is *true*.

Required Response Obligation Trigger

This obligation handler is triggered if the PDP response contains the obligation `http://glite.org/xacml/obligation/local-environment-map`, or the value defined by the *handledObligationId* parameter.

Required Request Attributes

This obligation handler **requires** the following request attributes in order to correctly map the user. The attributes can be provided by the gLite Grid Authorization Profile PIP, or directly by the clients implementing the XACML Grid Worker Node Authorization Profile (v.1.0) or the XACML Grid Computing Element Authorization Profile (v.1.0) specifications.

- The Subject Identifier
  - **type:** Subject
  - **id:** urn:oasis:names:tc:xacml:1.0:subject:subject-id
  - **data type:** urn:oasis:names:tc:xacml:1.0:data-type:x500Name
  - **multiple values allowed:** no
  - **description:** This is the Subject DN as given in the end-entity certificate. It is in RFC2253 format.

- The VOMS Primary FQAN
    - **type:** Subject
    - **id:** http://glite.org/xacml/attribute/fqan/primary
    - **data type:** http://glite.org/xacml/datatype/fqan
    - **issuer:** DN of the attribute certificate issuer
    - **multiple values allowed:** no
    - **description:** The primary Fully Qualified Attribute Name (FQAN) for the subject

- The VOMS FQANs
    - **type:** Subject
    - **id:** http://glite.org/xacml/attribute/fqan
    - **data type:** http://glite.org/xacml/datatype/fqan
    - **multiple values allowed:** yes
    - **description:** All the Fully Qualified Attribute Name (FQAN)s for the subject

Response Obligation Results

This result of this obligation handler is the **replacement** of the generic `http://glite.org/xacml/obligation/local-environment-map` with the more specific obligation `http://glite.org/xacml/obligation/local-environment-map/posix`.

This later obligation carries the account and group names in the following attribute assignments:

- one `http://glite.org/attribute/xacml/user-id` account login name
- zero or one `http://glite.org/attribute/xacml/group-id/primary` primary group name
- zero or more `http://glite.org/xacml/attribute/group-id` secondary group names

Example Configuration

This is an example PEPd configuration file with one gridmap POSIX account mapping obligation handler defined:

```
[SERVICE]
entityId = http://argus.example.org/pep
hostname = argus.example.org
obligationHandlers = ACCOUNT_MAPPING_OH

[PDP]
pdps = https://argus.example.org:8152/authz

[ACCOUNT_MAPPING_OH]
parserClass = org.glite.authz.pep.obligation.dfpmap.DFPMObligationHandlerConfigurationParser
accountMapFile = /etc/grid-security/grid-mapfile
groupMapFile = /etc/grid-security/group-mapfile
gridMapDir = /etc/grid-security/gridmapdir
```

Account and Group Mapping

This Obligation Handler uses the following logic to determine the mapping of the subject to a POSIX account.

Preconditions

- The input to this process is the subject DN of the end-entity certificate of the user and optionally a primary FQAN and a list of secondary FQANs.
- When dealing with the account map file (the gridmap file) and the group map file, entries are

Required Request Attributes

evaluated in the order listed in the file. Once a match is found processing stops.

- The grid map directory is populated with information for all configured pool accounts on the system. A pool account is considered "configured" if there is a zero-byte file, whose name is the pool account name, in the grid map directory. The grid map directory must also be read/writable by the user running the authorization service.

Mapping Steps

1. If a primary FQAN is given it is checked against the mappings listed in the account map file. If the primary FQAN matches a key in the map file then the associated value provides the account indicator.
2. If no account indicator was determined by means of the primary FQAN the subject DN is checked against the mappings listed in the account map file. If the DN matches a key in the map file then the associated value provides the account indicator. If no match is found processing stops and no map is available.
3. If the account indicator starts with a period ('.'), its value, without the period, is considered to be a pool account name prefix. If the account indicator does not start with a period it is a POSIX account name. If no account indicator was determined the mapping process fails.
4. If a primary FQAN is given then it is evaluated against entries in a group map file. The first entry that matches determines the primary group name. If no match is found, the mapping process fails.
5. If one or more secondary FQANs are given then each one is matched against the group map file and each match determines a secondary group name. If no matches occur then there are no secondary group names associated with the account.
6. If the account indicator is a POSIX account name, and zero or one primary group and zero or more secondary group names were determined then the mapping is completed. The user is mapped to that account.
7. If the account indicator is a pool account name prefix a lookup in the grid map directory occurs. The file looked for is generated according to the template `encoded_dn{:primary_group_name{:secondary_group_name}*}?` with the secondary group names listed in ascending alphabetical order.
8. If the file exists and has a link count of 2 then the hard link is followed to a file whose name is used as the POSIX account name. The last modified time of the two files is updated to the current time. If the link count is not 2 or the POSIX account name does not start with the account name prefix, the mapping process fails.
9. If the file does not exist, a list of files, within the grid map directory, matching the pool account name prefix followed exclusively by one or more numeric digits, is retrieved. An example regular expression representation of this would be `prod[0-9]+`
10. This list of files is searched for a file whose link count is 1, this is a candidate pool account file. A hard link to the pool account file, whose name corresponds to the filled in template described above is then created. If no such pool account file is found the mapping process fails.
11. The pool account file link count is rechecked, if it is more than 2 (indicating another request mapped to that same file at the same time) the created link is removed and the mapping process reverts to the previous step.
12. If the link count is 2 the mapping is complete. The user is mapped to the pool account corresponding to the given file.

Note: In the case where no FQANs are available, this obligation handler only returns a login name. No group information is returned.