



Universitatea "POLITEHNICA" București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Teză de doctorat

Controlul accesului în ATLAS TDAQ Online Cluster

Doctoral Thesis

Access Control in the ATLAS TDAQ Online Cluster

Doctorand: **Ing. Marius Constantin LEAHU**
Conducător științific: **Prof. Dr. Ing. Dan Alexandru STOICHESCU**
Prof. Dr. Ing. Vasile BUZULOIU

2013



Abstract

ATLAS (A Toroidal LHC Apparatus) is a general-purpose detector for studying high-energy particle interactions: it is the largest particle detector experiment at CERN and it is built around one of the interaction points of the proton beams accelerated by the Large Hadron Collider (LHC). The detector generates an impressive amount of raw data: 64 TB per second as a result of 40 MHz proton-proton collision rate with 1.6 MB data for each such event. The handling of such data rate is managed by a three levels Trigger and Data Acquisition (TDAQ) system, which filters out the events not relevant from physics research point of view and selects in the end in the order of 1000 events per second to be stored for offline analyses. This system comprises a significant number of hardware devices, software applications and human personnel to supervise the experiment operation. Their protection against damages as a result of misuse and their optimized exploitation by avoiding the conflicting accesses to resources are key requirements for the successful running of ATLAS. At the same time the number of users accessing the experiment resources from CERN and external institutes is considerable: the experiment is a collaboration involving roughly 3,000 physicists at 174 institutions in 38 countries. Additionally, the users are characterized by a high mobility between presence on site and at home universities locations. All these operation conditions call for an access control mechanism to protect the ATLAS resources.

This thesis presents our contribution to the analysis, design, implementation and deployment of the access control solution for the protection of ATLAS Online cluster and the TDAQ software running on it. The authors were involved in the research activity at CERN from 2004 to 2008 in the ATLAS System Administration team and the TDAQ Controls and Configuration team.

The access control solution we worked on is a step forward from the model based on group accounts used in past experiments at CERN to the model characterized by individual user accounts and permissions assignment to users by means of roles and roles hierarchy. Hence the access control solution for the ATLAS Online cluster revolves around the Role Based Access Control (RBAC) model which fulfills the ATLAS experiment's requirements for action traceability and accountability and offers the flexibility to accommodate the high number of users.

The original contribution of this thesis consists in designing a solution on top of RBAC model to address in a coherent way the protection needs from the cluster system administration level (remote access, login on the nodes, restrict access to tools execution on the nodes) to the TDAQ software level (TDAQ components protecting their functions). At the same time, the solution is open for integration with other experiment systems through the command line client and Application Programming Interface offered in Java and C++. Our work focuses on the authorization of user actions based on the access control policies, while the user authentication function is handled by the system administration specific services. The solution applies the RBAC concepts at system administration level with Linux traditional security mechanisms for seamless integration in the Scientific Linux CERN running on the cluster nodes. At the application level, we developed a dedicated service (TDAQ Access Manager) to serve the TDAQ Software components in managing the access control policies and to take the authorization decisions. We built this service on top of the OASIS XACML industry standard while paying special attention to the critical non-functional aspects like availability, performance, scalability and monitoring.

We finished the deployment in production in time for the first beam accelerated in LHC in autumn 2008. The setup currently consists in a high availability cluster of 6+1 nodes running the TDAQ Access Manager Service for ~3800 user accounts and ~440 roles. Each node of Access Manager Service is able to handle ~800 authorization requests per second from TDAQ software running on the ~3000 nodes of the ATLAS Online cluster. It is integrated with the system administration monitoring system for continue surveillance of service availability and performance. This production setup has run successfully in the last 4 years and has allowed ATLAS to take data steadily and efficiently, leading to the first major discovery: the Higgs boson.

Acknowledgments

Completing my PhD was a true journey and I would like to acknowledge the special persons who kindly supported me along this way.

It started thanks to my Professor Vasile Buzuloiu who encouraged me to pursue the research activity at CERN and supervised me since then. He sadly passed away before he could see this thesis written. Thank you Profu'!

I am grateful to Professor Dan Alexandru Stoichescu who “adopted” me at this stage and helped me to reach the finishing line.

I would like to thank the members of the ATLAS Trigger DAQ community at CERN, especially the two groups I was member of: the ATLAS TDAQ SysAdmins, and Controls and Configuration. The long discussions I had with Marc Dobson helped me to clarify the SysAdmin part of my work.

I am grateful to Giovanna Lehmann Miotto, the leader of Controls and Configuration group, for her guidance on the RBAC topic and driving the access control solution to a wider scope in the ATLAS experiment. Also a special thank you for Giovanna for her support and review while drafting this thesis.

I enjoyed a lot the company of Romanian fellows at CERN with whom I spent many joyful moments during my stay abroad. My colleagues from LAPI helped me a lot each time I had to take exams and dissertations. Thank you to all of you!

I thank my parents for their loving support and education, and my brother for being next to me during all these PhD years.

And most of all, I thank my wife for all her support during last years and to my boy for encouraging me with his first smiles while I was writing this thesis.

Contents

1. Introduction	1
1.1 About CERN	1
1.2 The ATLAS experiment	2
1.3 The ATLAS Trigger and Data Acquisition system	3
1.4 The Access Management in the Online Software	5
1.5 Thesis contribution and outline	6
1.5.1 Outline	7
2. Information security and access control models in software systems	9
2.1 Information security concepts	9
2.1.1 Threats	10
2.1.2 Goals of Security	12
2.1.3 Requirements	13
2.1.4 Security mechanisms	13
2.2 Design principles of security	14
2.2.1 Least Privilege	14
2.2.2 Fail-Safe Defaults	14
2.2.3 Economy of Mechanism	14
2.2.4 Complete Mediation	15
2.2.5 Open Design	15
2.2.6 Separation of Privilege	15
2.2.7 Least Common Mechanism	15
2.2.8 Psychological Acceptability	16
2.3 Access control concepts	16
2.3.1 Authorization versus authentication	16
2.3.2 Users, subjects, objects, operations and permissions	16
2.3.3 Policy, models and mechanisms	17
2.3.4 Reference monitor	18
2.4 Access control models	19
2.4.1 Discretionary Access Control	19
2.4.1.1 Access Control Lists	20
2.4.2 Mandatory Access Control	21
2.4.2.1 Bell-Lapadula model	22
2.4.3 Role Based Access Control	22
2.4.3.1 Brief History	22
2.4.3.2 Model overview	23
2.4.3.3 Flat RBAC	24
2.4.3.4 Hierarchical RBAC	25
2.4.3.5 Constrained RBAC	29
2.4.3.6 Symmetric RBAC	30
2.5 Conclusions	31
3. Access control solution for ATLAS Online computing system	33
3.1 Background and scope	33
3.1.1 Previous work	34
3.2 High level design	35

3.2.1	Access Control System Architecture	35
3.2.1.1	RBAC Administration	36
3.2.2	Access Control Enforcement	39
3.2.2.1	System Administration	39
3.2.2.2	Control Room Desktop	41
3.2.2.3	TDAQ Access Manager	41
3.2.3	Data flow	42
3.2.3.1	OASIS XACML perspective	42
3.2.3.2	Reference monitor perspective	45
3.3	Security design principles coverage	46
4.	<i>RBAC setup and administration</i>	47
4.1	Database type	47
4.2	RBAC setup	47
4.2.1	OpenLDAP service configuration	50
4.3	RBAC administration tool	50
4.3.1	Requirements	50
4.3.1.1	Administrative functions	51
4.3.1.2	User session management	53
4.3.1.3	Review functions	54
4.3.2	User accounts	54
4.3.3	Tools	55
4.3.3.1	Roles management	55
4.3.3.2	User's roles management	59
4.3.3.3	Permissions management	60
4.3.4	Requirements coverage matrix	61
5.	<i>Access control at the system administration level</i>	65
5.1	Protection of entry points into the ATLAS Online cluster	68
5.1.1	Remote access	68
5.1.1.1	Permissions	70
5.1.2	Control room desktops	72
5.1.2.1	Permissions	72
5.2	Protection on cluster nodes	73
5.2.1	Login restrictions	73
5.2.1.1	Permissions	74
5.2.2	Access control to sensitive tools	76
5.2.2.1	Permissions	76
6.	<i>Access control at the TDAQ Online Software level</i>	79
6.1	Requirements	80
6.1.1	Assumptions	80
6.1.2	Functional requirements	80
6.1.3	Non-functional requirements	81
6.2	Design	82
6.2.1	Policy Administration Point	84
6.2.1.1	XACML language for policy storage	86
6.2.2	Client-server model	88
6.2.2.1	Server	89
6.2.2.2	Client API	90
6.2.2.3	Non-functional aspects	93
6.2.3	Requirements coverage	95

6.3	Implementation.....	96
6.3.1	Policy Administration Point	96
6.3.2	Server	99
6.3.2.1	Authorization requests listener	99
6.3.2.2	Controller interface and statistics collector	102
6.3.2.3	Authorization logger.....	102
6.3.2.4	Scripts	103
6.3.3	Client API.....	105
6.4	Tests	108
6.4.1	Functional Tests	108
6.4.2	Performance and stress tests	108
6.5	Production Setup.....	115
6.5.1	AM server installation.....	116
6.5.2	Monitoring	117
6.5.3	Notifications from LDAP server.....	118
6.5.4	Client configuration	119
7.	Conclusions.....	121
7.1	Summary of contributions	121
7.2	Future work.....	122
8.	Appendices	125
8.1	LDAP schema for AM Roles	125
8.2	RBAC Administration Tool – User Requirements Document	127
8.3	Roles Manager shell script	128
8.4	Roles Display PHP script.....	135
8.5	User’s roles management shell script	140
8.6	Login restriction enforcement shell script.....	148
8.7	XACML policies generation by PAP	154
8.7.1	Input policies.....	155
8.7.2	(PP) Permission Policies for rules.....	156
8.7.2.1	crd.....	156
8.7.2.2	DAQ	159
8.7.2.3	Remote Access.....	167
8.7.3	(PPSrules) Permissions Policies Sets for rules	167
8.7.3.1	crd.....	167
8.7.3.2	DAQ	168
8.7.3.3	Remote Access.....	171
8.7.4	(RPS) Role Policies Sets	172
8.7.5	(PPSroles) Permissions Policies Sets for roles	175
8.8	Server statistics log sample	177
9.	Bibliography	179

List of figures

Figure 1 Overall view of LHC with its four main experiments.	2
Figure 2 Architectural view of the ATLAS detector.	3
Figure 3 Diagram of principal components of TDAQ system.	4
Figure 4 Classes of threats and CIA specific threats.	11
Figure 5 Reference monitor.	18
Figure 6 Discretionary Access Control using Access Control Lists	20
Figure 7 Mandatory Access Control	21
Figure 8 Hierarchical components in MAC.	22
Figure 9 Role Based Access Control.	23
Figure 10 Flat RBAC.	24
Figure 11 Hierarchical RBAC.	26
Figure 12 Example of role hierarchies.	27
Figure 13 Example of limited inheritance.	28
Figure 14 Constrained RBAC – Static SOD.	29
Figure 15 Constrained RBAC – Dynamic SOD.	30
Figure 16 Symmetric RBAC – Static SOD.	31
Figure 17 Symmetric RBAC – Dynamic SOD.	31
Figure 18 Access control software entities in the ATLAS online computing system.	35
Figure 19 The organizational and functional roles	37
Figure 20 Task permissions aggregation into intermediate non assignable roles	38
Figure 21 Example of roles hierarchy in ATLAS.	39
Figure 22 The ATLAS system architecture.	40
Figure 23 Layered access control on Linux nodes.	41
Figure 24 Examples of Access Manager clients.	42
Figure 25 Data flow diagram in the XACML standard.	43
Figure 26 Data flow in the ATLAS access control solution.	44
Figure 27 User definition in LDAP.	48
Figure 28 Roles and role hierarchy definitions in LDAP	48
Figure 29 Administrative Component use cases - Roles Management.	51
Figure 30 Administrative Component use cases - Users, Permissions Assignments to Roles.	52
Figure 31 Administrative Component use cases - Permissions.	52
Figure 32 Administrative Component use cases - Separation of Duties	53
Figure 33 Use Case diagram for Users Sessions management.	53
Figure 34 Use Case Diagram for Review Component.	54
Figure 35 The role tree for TDAQ:expert role.	58
Figure 36 The role tree for ShiftLeader role.	58
Figure 37 The role tree for DCS:expert role.	58
Figure 38 ATLAS Control Room.	66
Figure 39 Sequence diagram of how users the ATLAS Online cluster.	66
Figure 40 Access control enforcements at system administration level.	67
Figure 41 Centralized configuration of AM authorization for ATLAS Application Gateways ...	69
Figure 42 AM authorization disabled on the application gateway.	69
Figure 43 AM authorization enabled on the application gateway.	70
Figure 44 <i>RemoteAccess</i> role inherited by <i>TDAQ:expert</i>	71
Figure 45 Choose roles when logging on CRD	72
Figure 46 “Authentication Configuration” in SLC 5	74
Figure 47 Example of login restriction in LDAP	75
Figure 48 Example of sudo role in LDAP	77
Figure 49 TDAQ AM Service use cases	81

Figure 50 Mapping of XACML actors to AM service components	83
Figure 51 XACML policy language model	87
Figure 52 Hierarchical RBAC profile of XACML	88
Figure 53 AM Server high level design	90
Figure 54 AM Java client API.....	91
Figure 55 AM C++ client API – server interrogation	91
Figure 56 AM C++ client API – resource types.....	92
Figure 57 Example of how to use AM C++ client API.....	93
Figure 58 Retry mechanism in client API.....	94
Figure 59 Access Manager Policy Administration Point – Component diagram	97
Figure 60 Sequence diagram for XACML policies generation in PAP	97
Figure 61 Build and deployment of XACML policies to AM servers	98
Figure 62 Reactor design pattern in AM server.....	100
Figure 63 Access Manager server - Component diagram.....	101
Figure 64 Component diagram of AM Java client API	107
Figure 65 Component diagram of AM C++ client API.....	107
Figure 66 Histogram of first performance tests (2 nd run).....	113
Figure 67 Histogram of second performance tests	114
Figure 68 Production setup for TDAQ AM Service	116
Figure 69 AM server configuration in LDAP	117
Figure 70 AM server monitoring	118
Figure 71 Notifications from LDAP servers to AM servers	119
Figure 72 Sample roles hierarchy in LDAP.....	154

List of tables

Table 1 RBAC variations organized as levels.....	24
Table 2 Users' frequently asked question on the access control in ATLAS experiment.....	34
Table 3 Mapping of Reference Monitor concept to XACML model	45
Table 4 Security design principles coverage.....	46
Table 5 Roles definition in LDAP.....	49
Table 6 Permissions for remote access	71
Table 7 Permissions for CRD.....	73
Table 8 The components of a permission from PAP input file	84
Table 9 Process Manager resource category	85
Table 10 Run Control resource category	85
Table 11 IGUI resource category	85
Table 12 Resource Manager resource category	85
Table 13 Data Base resource category	86
Table 14 BCM resource category.....	86
Table 15 Non-functional requirements coverage by TDAQ AM.....	95
Table 16 AM server configuration on the first set of performance tests (2 nd run)	111
Table 17 Client configuration on the first set of performance tests (2 nd run).....	112
Table 18 First performance test results (comparison between runs).....	112
Table 19 AM server configuration on the second set of performance test	113
Table 20 Second performance test results.....	114
Table 21 Environment variables for client API	119

Chapter 1

Introduction

This thesis presents the contributions of the author to the conception, development and deployment in production of the access control system for the ATLAS experiment's computing cluster and data acquisition software running on it. This work has been done during the years spent by the author at CERN [1] as part of the ATLAS TDAQ SysAdmin group [2] and ATLAS TDAQ Controls and Configuration group.

1.1 About CERN

CERN, the European Organization for Nuclear Research, was founded in 1954 and is one of the world's largest and most respected centers for scientific research. CERN's main function is to provide the particle accelerators and other infrastructure needed for high-energy physics research. It is also noted for being the birthplace of the World Wide Web. The CERN Laboratory sits astride the Franco-Swiss border near Geneva. It was one of Europe's first joint ventures and now has 20 Member States, and maintains collaborations with over 200 institutes and universities from non-member states.

Most of the activities at CERN are currently directed towards operating the new Large Hadron Collider (LHC), and the experiments for it. The LHC start-up event [3] took place on September 10, 2008, with a pause for maintenance until November 20, 2009 when the operation resumed [4].

The LHC represents a large-scale, worldwide scientific cooperation project and is the world's largest and most powerful particle accelerator. It mainly consists of a 27 km ring of superconducting magnets with a number of accelerating structures to boost the energy of the particles along the way (Figure 1).

Inside the accelerator, two beams of particles of the same kind (either protons or lead ions) travel at close to the speed of light with very high energies before colliding with one another. The beams travel in opposite directions in separate beam pipes – two tubes kept at ultrahigh vacuum. They are guided around the accelerator ring by a strong magnetic field, achieved using superconducting electromagnets. These are built from coils of special electric cable that operates in a superconducting state, efficiently conducting electricity without resistance or loss of energy. This requires chilling the magnets to about -271°C – a temperature colder than outer space!

Each proton beam flying around the LHC will have an energy of 7 TeV, so when two proton beams collide the collision energy will be 14 TeV. Lead ions have many protons, and together they give an even greater energy: the lead-ion beams will have a collision energy of 1150 TeV. Both collision energies have never been reached before in a lab. Energy concentration is what makes particle collisions so special. In absolute terms, these energies, if compared to the energies we deal with everyday, are not impressive. In fact, 1 TeV is about the energy of motion of a flying mosquito. What makes the LHC so extraordinary is that it squeezes energy into a space about a million times smaller than a mosquito.

In the LHC, under nominal operating conditions, each proton beam has 2808 bunches, with each bunch containing about 10^{11} protons. At full luminosity the LHC uses a bunch spacing of 25 ns

(or about 7 m). This corresponds to a frequency of 40 MHz, which implies that bunches should pass each of the collision points in the LHC 40 million times a second.

The most comprehensive model of particle interactions available today is known as the Standard Model. With the important exception of the Higgs boson, all of the particles predicted by the model have been observed. This model however, does not apply to energies greater than 1 TeV. As the LHC will work at 14 TeV, it is hoped that the experiments will provide enough data for physicists to develop a new theory that applies to higher energies and is able to predict the particles that can appear with all their properties: masses, momenta, energies, charges and nuclear spins.

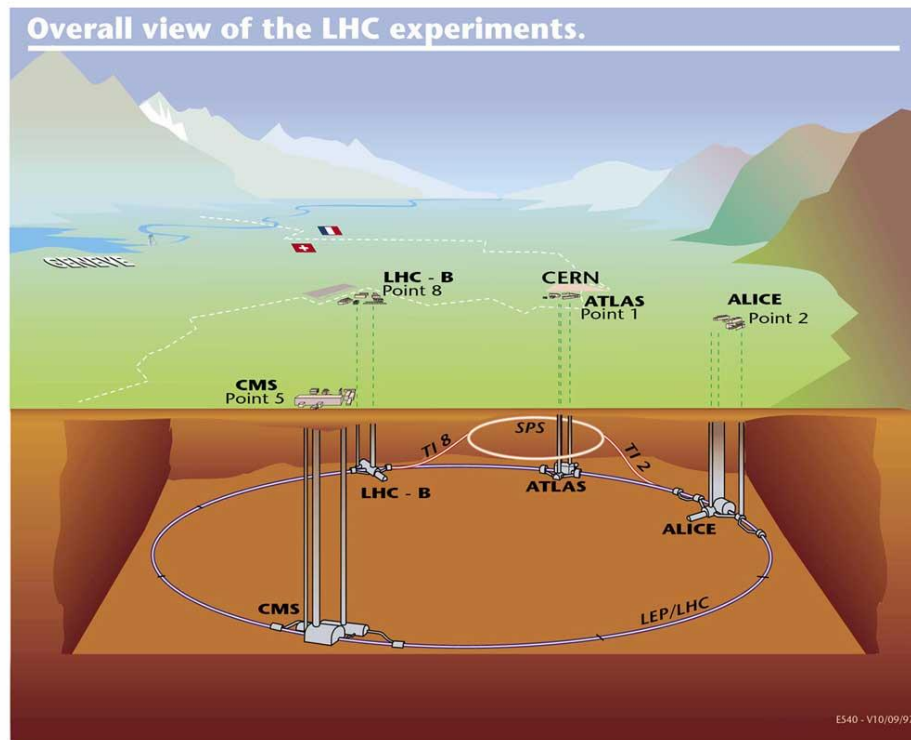


Figure 1 Overall view of LHC with its four main experiments.

The six experiments at the LHC are all run by international collaborations, bringing together scientists from institutes all over the world. Each experiment is distinct, characterized by its unique particle detector.

The two large experiments, ATLAS and CMS, are based on general-purpose detectors to analyze the myriad of particles produced by the collisions in the accelerator. They are designed to investigate the largest range of physics possible. Having two independently designed detectors is vital for cross-confirmation of any new discoveries made.

Two medium-size experiments, ALICE and LHCb, have specialized detectors for analyzing the LHC collisions in relation to specific phenomena.

Two experiments, TOTEM and LHCf, are much smaller in size. They are designed to focus on 'forward particles' (protons or heavy ions). These are particles that just brush past each other as the beams collide, rather than meeting head-on.

1.2 The ATLAS experiment

The ATLAS experiment (A Toroidal LHC ApparatuS) [5] [6] is a general-purpose experiment for recording proton-proton collisions at LHC. It is conducted by a collaboration of 3000 scientists from 174 universities and laboratories representing 38 countries around the world.

The ambitious experimental program of ATLAS will shed light on many unanswered questions about the origins of matter and the fundamental forces of nature. ATLAS is intended to investigate many different types of physics that might become detectable in the energetic collisions of the LHC. Some of these are confirmations or improved measurements of the Standard Model, while many others are searches for new physical theories. One of the most important goals of ATLAS is to investigate a missing piece of the Standard Model, the Higgs boson [7]. *Sooner than expected, the Higgs search has shown its first successful results [8]: the Higgs boson has been observed and now its properties are to be measured and predictions to be verified.*

The ATLAS detector (Figure 2) is not only complex but also very big – measuring 46 m long and 25 m high, it will be the largest-volume detector ever constructed for particle physics. The head-on collisions of protons at its centre leave debris that will reveal new particles and new processes in the interior of matter. The detector consists of several concentric cylindrical layers built around the LHC beam intersection point. The layers are composed of various sub-detectors from the three major detection systems:

- Inner Detector measures the paths of electrically charged particles and is composed of three parts: Pixel, Silicon Tracker (SCT), and Transition Radiation Tracker (TRT).
- Calorimeters measure the energy of charged or neutral particles (Tile and Liquid Argon).
- Muon Spectrometer detects muons, heavy particles that cannot be stopped by the calorimeters.

The curvature of particle tracks in the magnetic field will allow the momentum and electric charges to be determined. Out of nearly 1000 million collisions each second, only a few will have the special characteristics that might lead to new discoveries. The trigger system selects such events for recording and avoids storing immense amounts of unnecessary information.

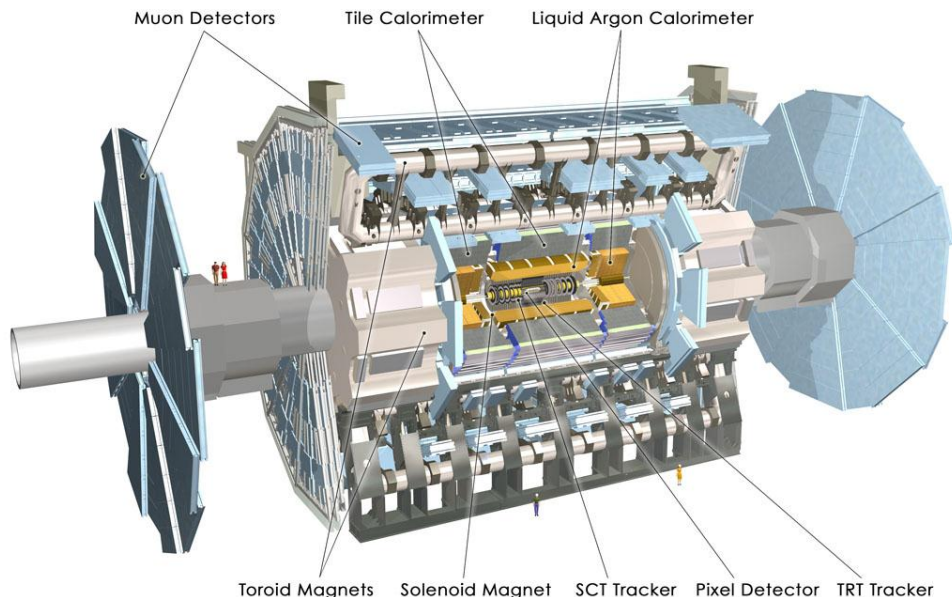


Figure 2 Architectural view of the ATLAS detector.

1.3 The ATLAS Trigger and Data Acquisition system

With a LHC bunch crossing rate of 40 MHz and about 23 interactions per bunch crossing, a highly selective trigger system to reduce the expected 10^9 interactions per second to an acceptable rate of a few hundred Hz is required. The Trigger and Data Acquisition (TDAQ) [9] system is

responsible for the filtering and the preparation for archival of the events captured by the ATLAS detector.

The ATLAS trigger is based on three levels of online event selection [10]: Level-1, Level-2, and Event Filter (EF). While the Level-1 trigger is implemented in custom hardware, the second and third level triggers (together known as the High Level Trigger (HLT)) are software based and implemented on personal computers (PC) running the Linux operating system.

The **Level-1 trigger** reduces the initial event rate of 40 MHz to about 75 kHz as shown in Figure 3 [9]. During the latency of the L1 trigger selection algorithms (up to 2.5 μ s), the complete event data is kept in the pipeline memories of the detector front-end electronics. Only the data for the events selected by the L1 trigger is then transferred from these front-end memories into the Read-Out Buffers (ROBs) contained in the **Read-Out System** units (ROSs), where it is temporarily stored and provided on request to the following stages of event selection. The data from the large number of detector readout channels is multiplexed into 1600 data fragments by the detector-specific Read-Out Drivers (RODs) and each of these fragments is sent for storage to an individual ROB. For every accepted event, the L1 system produces the "Region of Interest" (RoI) information, which includes the positions of all the identified interesting objects in units of pseudo-rapidity and azimuthal angle. This information is sent by the different elements of the L1 trigger system to the RoI builder (ROIB), which assembles it into a unique data fragment and sends it to a Level 2 supervisor (L2SV).

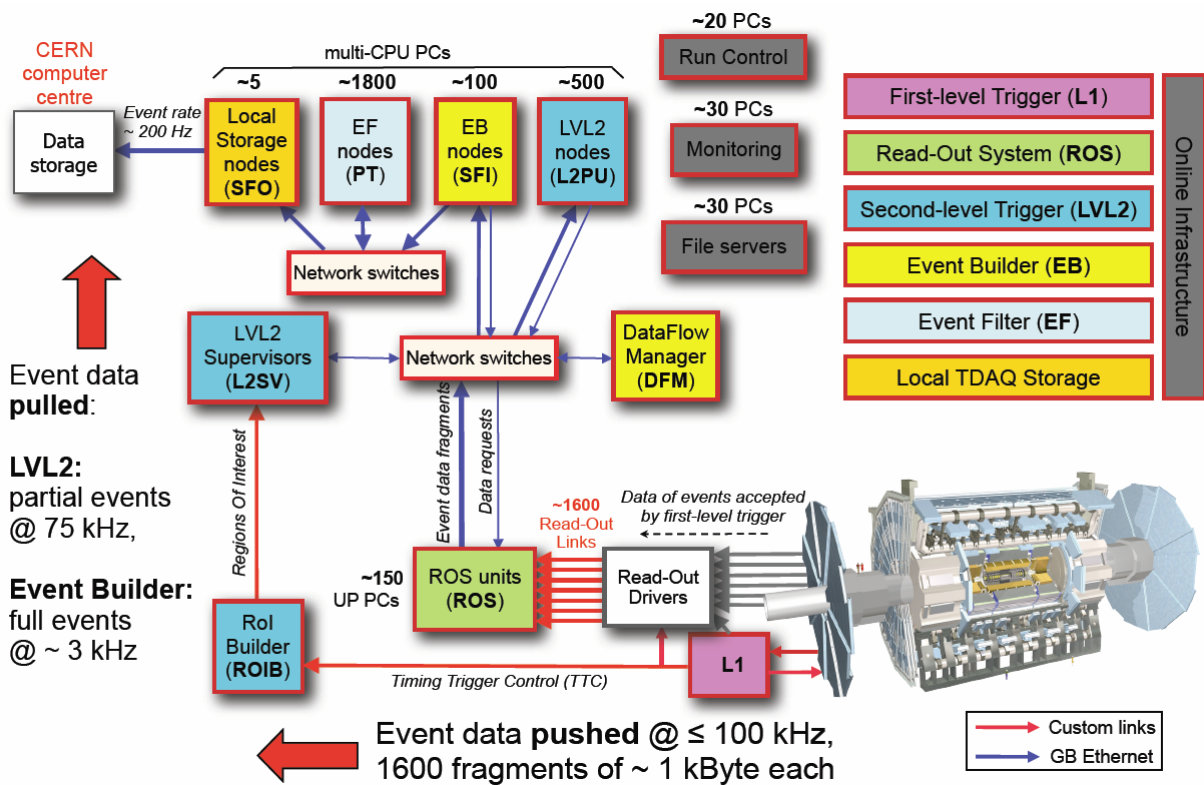


Figure 3 Diagram of principal components of TDAQ system.

The L2SVs are the steering elements of the **second trigger level (L2)**, which is designed to provide an additional factor 20-30 in reduction power. They receive the RoI information, assign the events to one of the processing units (L2PUs) running on a L2 node, and handle the results of the selection algorithms. To provide the requested reduction power, the L2PUs need to access detailed information from all the ATLAS detector elements (muon system, calorimeters and inner detector). To minimize the data transfers required at this early stage, the L2PUs retrieve only the few data fragments related to the geographical addresses of the interesting objects identified by the L1 (1-2 % of the total data volume). To do so it uses the RoI information received by the L2SV to identify and

access only the few ROBs containing the relevant data fragments. A fast identification of the relevant ROBs is made possible by the fact that there is simple and fixed correspondence between the RoI regions and the ROBs, as each of them always receive data fragments from the same specific detector front-end modules. The L2 system is really the most characteristic element of the ATLAS architecture, and provides detailed selection power before the full event-building and consequently reduces the overall dataflow power needs.

The results of the L2 algorithms are sent by the L2SVs to the data flow manager (DFM), which assigns the accepted events to the **event building** nodes (SFIs) according to load-balancing criteria. The SFIs collect the data fragments related to any assigned event from all the ROBs and assemble them in a unique event fragment. The expected rate of events at this stage is 3.5 kHz, which, given a mean ATLAS event size of 1.6 MByte, corresponds to a total throughput of about 6 GByte/s out of the event building system.

The resulting complete event fragments are then sent to the **event filter** nodes, where they are assigned to a processing task (PT) running on an Event Filter Processor (EFP) for the last selection stage. The accepted events are finally sent to the output nodes (SFOs) to be temporarily buffered on local disks and transferred to the CERN computing center for permanent recording on mass storage. At this stage the rate of events is expected to be 0.2 kHz, i.e. more than a factor 105 lower than the original LHC bunch-crossing rate.

The DFM also manages the list of events that can be removed from the data flow system, as they have either been rejected by the L2 or received by an EFP, and periodically sends to the ROBs the list of data fragments to be released.

The Online Software system which runs on the **Online Infrastructure** is responsible for configuring, controlling, and monitoring the TDAQ system, but excludes any management, processing, or transportation of event data. It is a framework which provides the glue between the above trigger levels and Detector Control System (DCS), and defines interfaces to those elements. It also includes information-distribution services and access to configuration and other meta-data databases. An important function of the Online Software is to provide the services that enable the TDAQ and detector systems to start up and shut down. It is also responsible for the synchronization of the entire system, and the supervision of processes. Verification and diagnostics facilities help with the early detection of problems. The configuration services provide the framework for storing the large amount of information required to describe the system topology, including hardware and software components. During data taking, access is provided to monitoring tasks, histograms produced in the TDAQ system, and also the errors and diagnostics messages sent by different applications. One or more user interfaces display the available information and enable the user to configure and control the TDAQ system.

1.4 The Access Management in the Online Software

The **Online Software** architecture is based on a component model and consists of three high level components, called packages. Each of the packages is associated with a group of functions of the Online Software. For each package, a set of services which it provides are defined. The services are clearly separated one from another and have well defined boundaries. For each service a low-level component, called a sub-package, is identified.

Each package is responsible for a clearly defined functional aspect of the whole system.

- **Control** — contains sub-packages for the control of the TDAQ system and detectors. Control sub-packages exist to support TDAQ system initialization and shutdown, to provide control command distribution, synchronization, error handling, and system verification. The details on the evolution of this system are in [11].

- Databases — contain sub-packages for configuration of the TDAQ system and detectors. Configuration sub-packages exist to support system configuration description and access to it, record operational information during a run and access to this information. There are also boundary classes to provide read/write access to the conditions storage.
- Information Sharing — contains classes to support information sharing in the TDAQ system. Information Sharing classes exist to report error messages, to publish states and statistics, to distribute histograms built by the sub-systems of the TDAQ system and detectors, and to distribute events sampled from different parts of the experiment's data flow chain.

The **Access Management** is a general Online Software security service, responsible for TDAQ user authorization. It enforces an access policy, in order to stop non-authorized persons from corrupting the TDAQ system and interfering with data taking. This applies in particular to sensitive areas like the access to configuration databases, access to process management, remote access through the Web, etc. Apart from these cases, the limited access to the network at the experiment site is the main security measure.

1.5 Thesis contribution and outline

The main contribution of this thesis has been the design, implementation and deployment of a complete access control solution for the TDAQ Software and the ATLAS TDAQ computing environment.

The large scale of ATLAS collaboration (over 3000 people) with its significant experiment resources, both hardware and software, raises the exposure to the usual computing security threats which can cause dead time in the experiment operation and, in the end, financial losses. This issue has been addressed in the ATLAS TDAQ Control and Configuration system where a dedicated service, the Access Management, was put in charge with the software security. Since the protection is also necessary at the level of the computing cluster where the TDAQ software runs, and even to other systems in ATLAS, the need for a higher level approach on the software security solution emerged naturally. We have embraced the challenge of designing an access control solution to harmonize the protection at the operating system level on the cluster nodes with the Access Management functions.

Given the high number of people foreseen to work in the ATLAS experiment, the Role Based Access Control (RBAC) model [12] was chosen for its scalability, flexibility, ease of administration and usability from the lowest operating system level to the highest software application level. We have designed a centralized configuration of RBAC entities and policies, then the mechanisms to enforce automatically the access control policies on the cluster nodes by restricting the user's log in (locally or remotely) and execution of sensitive tools on the nodes.

We have designed the Access Manager service to use the RBAC centralized configuration and be compliant with the XACML (eXtensible Access Control Markup Language) standard [13] in defining and processing the access control policies. Implemented on a client-server model, the Access Manager server takes the authorization decisions when requested by its clients (other TDAQ systems) and the clients enforce the decision by allowing or denying user actions.

Finally, we carried out the implementation of the access control solution, the tests in the test laboratory cluster and the deployment in the end to the production cluster where it is currently running.

1.5.1 Outline

The following chapter introduces the information security concepts and principles that guide the design of a software security solution. The three most widely used access control models (Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC)) are described with their characteristics, and then the arguments for choosing RBAC are presented.

In Chapter 3 we present the background and scope of the access control in ATLAS experiment, and then the high level solution for access control is outlined in three parts: the RBAC administration, where the access control is enforced and what is the data flow in the authorization decision. The chapter ends with the coverage matrix of how the security design principles are addressed by our solution.

Starting with Chapter 4 we dive into the solution description. First, it is presented how the RBAC configuration is centralized in cluster's directory service (a Lightweight Directory Access Protocol (LDAP) [14] server) with the mapping of RBAC concepts to the LDAP structure. The tools developed to facilitate the RBAC management are also listed. The aspects concerning the access control at the operating system level are addressed in Chapter 5, while the Access Management system design and implementation with its integration in the Control and Configuration software are presented in detail in Chapter 6.

Chapter 7 concludes the work presented in this thesis and outlines the directions for further improvements of the access control solution and integration with other systems at CERN.

Chapter 2

Information security and access control models in software systems

The information is an important strategic and operational asset for any organization and a failure in protecting it from damages or misuse affects not only a single user or an application, but the organization itself with possible disastrous consequences. Moreover, the advent of Internet as well as networking capabilities has made the access to information much easier, therefore the control of the access to it represents the starting point in providing security for the organization. The demand for security safeguards has long been dominated by the military. As a result, the orientation is rather different from what corporations, government agencies and the public really need. Meanwhile, the supply of security safeguards has been dominated by computing and communications specialists. Hence the two most used access control models: Mandatory Access Control (MAC) appropriate for multilevel secure military applications and Discretionary Access Control (DAC) perceived as meeting the security processing needs of industry and civilian government. The Role Based Access Control (RBAC) model has emerged as a viable alternative to traditional DAC and MAC because it is based on an enterprise's organizational structure.

Although role-based security models have existed for 20 years, their application has until recently been limited. To date, most systems have based access control on the discretion of the owner or administrator of the data as opposed to basing access on organizational or policy needs as is done with RBAC. These owner-controlled systems worked adequately for small local area networks (LAN) but have become cumbersome to manage and errors prone as networking capabilities have increased. The explosion of electronic data exchange and interconnection of information systems led to significant productivity gains in the 1990s. However, these same factors have also increased electronic security and integrity concerns. Confidentiality restriction and regulatory requirements have caused organizations to look for improved approaches to manage the types of users that may have access to which data and to which applications. The result is a renewed and growing interest in role-based security models.

This chapter presents firstly the basic concepts of information security followed by the design principles of security and it ends presenting the DAC, MAC and RBAC with the focus on the last one.

2.1 Information security concepts

Information security deals with several different trust aspects of information. Information security is not confined to computer systems nor to information in an electronic or machine-readable form. It applies to all aspects of safeguarding or protecting information or data, in whatever form. One of the definitions of information security is given by [15]:

the protection of information systems against unauthorized access to or modification of information, whether in storage, processing or transit, and against the denial of service to authorized users or the provision of service to unauthorized users, including those measures necessary to detect, document, and counter such threats

One important point is that information security is not perfect. No one can ever eradicate all risk of improper or capricious use of any information, but the level of information security should be

commensurate with the value of the information and the loss, financial or otherwise, that might occur from improper use - disclosure, degradation, denial, or whatever.

2.1.1 Threats

A *threat* is a potential violation of security. The violation need not actually occur for there to be a threat. The fact that the violation might occur means that those actions that could cause it to occur must be guarded against (or prepared for). Those actions are called *attacks*. Those who execute such actions, or cause them to be executed, are called *attackers*. All the information security attributes are subjects of threats.

The threats can be classified in four categories:

- **disclosure**, or unauthorized access to information
- **deception**, or acceptance of false data
- **disruption**, or interruption or prevention of correct operation
- **usurpation**, or unauthorized control of some part of a system

Many common threats can be found in one of these classes. Following paragraphs will details a few of the common threats and Figure 4 classifies them in the classes of threats with respect to the CIA triad.

Snooping, the unauthorized interception of information, is a form of disclosure. It is passive, suggesting simply that some entity is listening to (or reading) communications or system information. Wiretapping, or **passive wiretapping**, is a form of snooping in which a network is monitored. (It is called "wiretapping" because of the "wires" that compose the network, although the term is used even if no physical wiring is involved.) Confidentiality requirement counter this threat.

Modification or alteration, an unauthorized change of information, covers three classes of threats. The goal may be deception, in which some entity relies on the modified data to determine which action to take, or in which incorrect information is accepted as correct and is released. If the modified data controls the operation of the system, the threats of disruption and usurpation arise. Unlike snooping, modification is active; it results from an entity changing information. **Active wiretapping** is a form of modification in which data moving across a network is altered; the term "active" distinguishes it from snooping ("passive" wiretapping). An example is the **man-in-the-middle** attack, in which an intruder reads messages from the sender and sends (possibly modified) versions to the recipient, in hopes that the recipient and sender will not realize the presence of the intermediary. Integrity requirement counter this threat.

Masquerading or spoofing, an impersonation of one entity by another, is a form of both deception and usurpation. For example, if a user tries to log into a computer across the Internet but instead reaches another computer that claims to be the desired one, the user has been spoofed. This may be a passive attack (in which the user does not attempt to authenticate the recipient, but merely accesses it), but it is usually an active attack (in which the masquerader issues responses to mislead the user about its identity). Although primarily deception, it is often used to usurp control of a system by an attacker impersonating an authorized manager or controller. Integrity requirement (called "authentication" in this context) counter this threat.

Some forms of masquerading may be allowed. **Delegation** occurs when one entity authorizes a second entity to perform functions on its behalf. The distinctions between delegation and masquerading are important. If Alice delegates to Bob the authority to act on her behalf, she is giving permission for him to perform specific actions as though she were performing them herself. All parties are aware of the delegation. Bob will not pretend to be Alice; rather, he will say, "I am Bob and I have authority to do this on Alice's behalf." If asked, Susan will verify this. On the other hand, in a masquerade, Bob will pretend to be Alice. No other parties (including Alice) will be aware of the masquerade, and Bob will say, "I am Alice." Should anyone discover that he or she is dealing with Bob

and ask Alice about it, she will deny that she authorized Bob to act on her behalf. In terms of security, masquerading is a violation of security, whereas delegation is not.

Repudiation of origin, a false denial that an entity sent (or created) something, is a form of deception. For example, suppose a customer sends a letter to a vendor agreeing to pay a large amount of money for a product. The vendor ships the product and then demands payment. The customer denies having ordered the product and by law is therefore entitled to keep the unsolicited shipment without payment. The customer has repudiated the origin of the letter. If the vendor cannot prove that the letter came from the customer, the attack succeeds. A variant of this is denial by a user that he created specific information or entities such as files. Integrity mechanisms cope with this threat.

Denial of receipt, a false denial that an entity received some information or message, is a form of deception. Suppose a customer orders an expensive product, but the vendor demands payment before shipment. The customer pays, and the vendor ships the product. The customer then asks the vendor when he will receive the product. If the customer has already received the product, the question constitutes a denial of receipt attack. The vendor can defend against this attack only by proving that the customer did, despite his denials, receive the product. Integrity and availability mechanisms guard against these attacks.

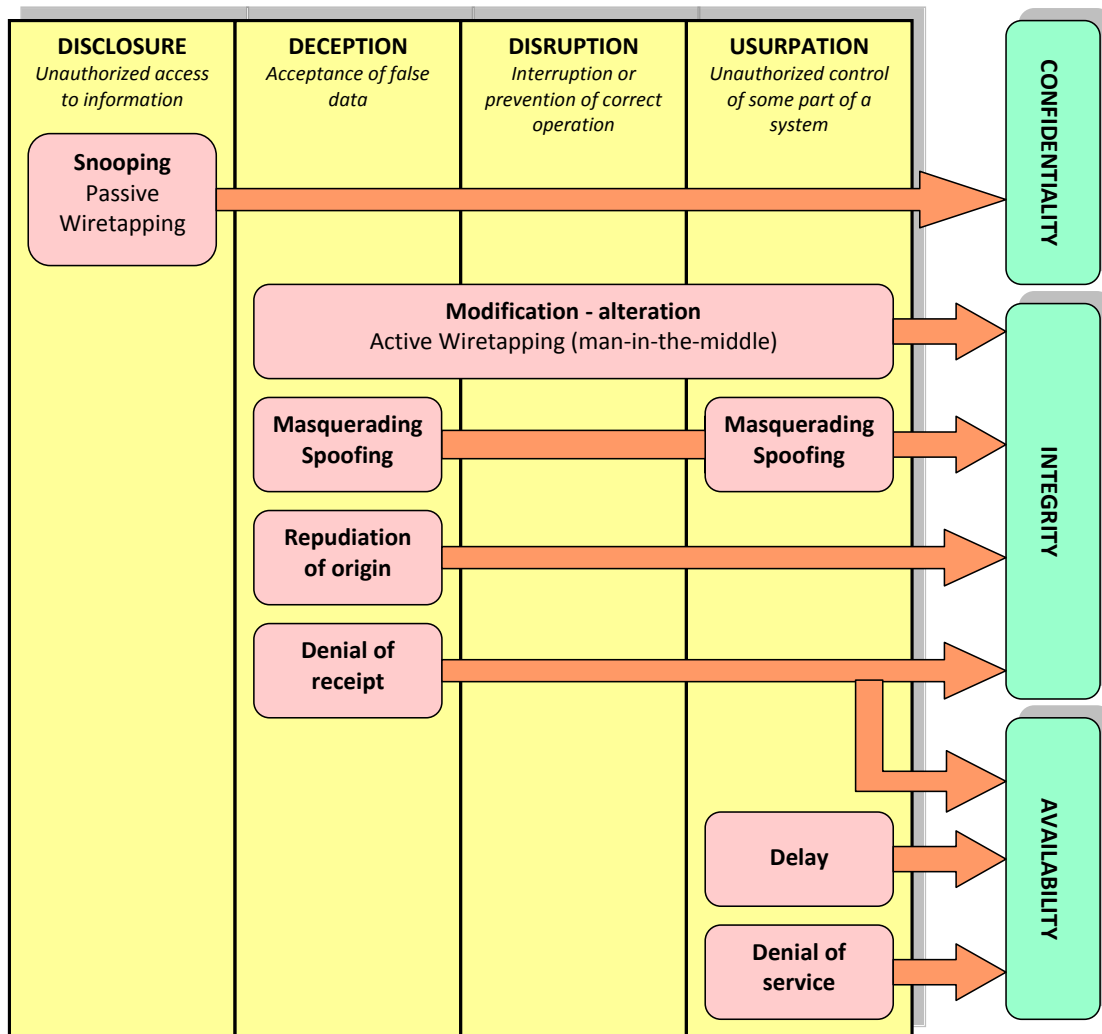


Figure 4 Classes of threats and CIA specific threats.

Delay, a temporary inhibition of a service, is a form of usurpation, although it can play a supporting role in deception. Typically, delivery of a message or service requires some time t ; if an

attacker can force the delivery to take more than time t , the attacker has successfully delayed delivery. This requires manipulation of system control structures, such as network components or server components, and hence is a form of usurpation. If an entity is waiting for an authorization message that is delayed, it may query a secondary server for the authorization. Even though the attacker may be unable to masquerade as the primary server, she might be able to masquerade as that secondary server and supply incorrect information. Availability mechanisms can thwart this threat.

Denial of service, a long-term inhibition of service, is a form of usurpation, although it is often used with other mechanisms to deceive. The attacker prevents a server from providing a service. The denial may occur at the source (by preventing the server from obtaining the resources needed to perform its function), at the destination (by blocking the communications from the server), or along the intermediate path (by discarding messages from either the client or the server, or both). Denial of service poses the same threat as an infinite delay. Availability mechanisms counter this threat.

Denial of service or delay may result from direct attacks or from non security-related problems. From our point of view, the cause and result are important; the intention underlying them is not. If delay or denial of service compromises system security, or is part of a sequence of events leading to the compromise of a system, then we view it as an attempt to breach system security. But the attempt may not be deliberate; indeed, it may be the product of environmental characteristics rather than specific actions of an attacker.

2.1.2 Goals of Security

Prior to the enumeration of security goals, it is needed to make a clear distinction between policy and mechanism.

A **security policy** is a statement of what is and what is not allowed.

A **security mechanism** is a method, tool or procedure for enforcing a security policy.

Policies may be presented mathematically, as a list of allowed (secure) and disallowed (nonsecure) states. In practice, the policies are rarely so precise; they normally describe in human language what users are allowed to do. The ambiguity inherent in such a description leads to states that are not classified as "allowed" or "disallowed".

Given a security policy's specification of "secure" and "nonsecure" actions, these security mechanisms can *prevent* the attack, *detect* the attack, or *recover* from the attack. The strategies may be used together or separately.

Prevention means that an attack will fail, i.e. security policy has not been violated. Typically, prevention involves implementation of mechanisms that users cannot override and that are trusted to be implemented in a correct, unalterable way, so that the attacker cannot defeat the mechanism by changing it. Preventative mechanisms often are very cumbersome and interfere with system use to the point that they hinder normal use of the system. Prevention mechanisms can prevent compromise of parts of the system; once in place, the resource protected by the mechanism need not be monitored for security problems, at least in theory.

Detection is most useful when an attack cannot be prevented, but it can also indicate the effectiveness of preventative measures. Detection mechanisms accept that an attack will occur; the goal is to determine that an attack is under way, or has occurred, and report it. The attack may be monitored, however, to provide data about its nature, severity, and results. Typical detection mechanisms monitor various aspects of the system, looking for actions or information indicating an attack. Detection mechanisms do not prevent compromise of parts of the system, which is a serious drawback. The resource protected by the detection mechanism is continuously or periodically monitored for security problems.

Recovery has two forms. The first is to stop an attack and to assess and repair any damage caused by that attack. In practice, recovery is very complex, because the nature of each attack is unique. Thus, the type and extent of any damage can be difficult to characterize completely. Moreover, the attacker may return, so recovery involves identification and fixing of the vulnerabilities used by the attacker to enter the system. In some cases, retaliation (by attacking the attacker's system or taking legal steps to hold the attacker accountable) is part of recovery. In all these cases, the system's functioning is inhibited by the attack. By definition, recovery requires resumption of correct operation.

In a second form of recovery, the system continues to function correctly while an attack is under way. This type of recovery is quite difficult to implement because of the complexity of computer systems. It draws on techniques of fault tolerance as well as techniques of security and is typically used in safety-critical systems. It differs from the first form of recovery, because at no point does the system function incorrectly. However, the system may disable nonessential functionality. Of course, this type of recovery is often implemented in a weaker form whereby the system detects incorrect functioning automatically and then corrects (or attempts to correct) the error.

2.1.3 Requirements

Three widely accepted elements of information security, referred to either as requirements, principles, qualities or attributes, are: confidentiality, integrity and availability, also remembered by the mnemonic **CIA** or CIA triad. A simple way to express this is "the right information to the right people at the right time". Other elements are: accountability and completeness.

Confidentiality includes restricting access to information to those who are privileged to see it. The term *privacy* is often used when information to be protected refers to individuals.

Integrity is trust that can be placed in the information. *Data integrity* is having trust that the information has not been altered between its transmission and its reception. Information has to be protected from unauthorized modifications or incorrect modifications (referred to as semantic integrity). *Source integrity* is having trust that the sender of that information is who it is supposed to be.

Availability defines that information or resources are available when required. Most often this means that the resources are available at a rate which is fast enough for the wider system to perform its task as intended.

Accountability, a "fourth component," is synonymous with non-repudiation. The non-repudiation of receipt of information means that an agent can't deny receiving information. This can prevent an online vendor from being obliged to ship replacement goods to a malicious customer who denies receiving the original items. The non-repudiation of sourcing information means that an agent can't deny sending information. This prevents an agent from anonymously sending spoofed emails with malicious intent, for example.

Completeness refers to ensure that subjects receive all information they are entitled to access, according to the stated security policies.

2.1.4 Security mechanisms

There are many types of mechanisms that aim to ensure the security characteristics of a particular system. Confidentiality is enforced by the access control mechanism. Integrity is enforced by the access control mechanism and by the semantic integrity constraints. Availability is enforced by the recovery mechanism and by detection techniques for DoS attacks – an example of which is query flood. Additional mechanisms are:

- user authentication: to verify the identity of subjects wishing to access the information

- information authentication: to ensure information authenticity; it is supported by signature mechanisms
- encryption: to protect information when being transmitted across systems and when being stored on secondary storage
- intrusion detection: to protect against impersonation of legitimate users and also against insider threats

2.2 Design principles of security

Saltzer and Schroeder [16] defined the 8 principles for the design and implementation of security systems. They are based on the ideas of simplicity and restriction.

Simplicity makes design and mechanisms easy to understand and less can go wrong with simple design. The number of inconsistencies that can occur is also minimized.

Restriction minimizes the power of an entity which can access only the information it needs. This is also known as “need to know” principle. By inhibition of communication, an entity can communicate with other entities only when necessary, and in a few ways as possible.

2.2.1 Least Privilege

This principle restricts how privileges are granted.

The principle of least privilege states that an entity should be given only those privileges that it needs in order to complete its task.

If a subject does not need an access right, the subject should not have that right. Furthermore, the function of the subject (as opposed to its identity) should control the assignment of rights. If a specific action requires that a subject's access rights be augmented, those extra rights should be relinquished immediately on completion of the action. This is the analogue of the "need to know" rule: if the subject does not need access to an object to perform its task, it should not have the right to access that object.

In practice, most systems do not have the granularity of privileges and permissions required to apply this principle precisely. The designers of security mechanisms then apply this principle as best they can. In such systems, the consequences of security problems are often more severe than the consequences for systems that adhere to this principle.

2.2.2 Fail-Safe Defaults

This principle restricts how privileges are initialized when a subject or object is created.

The principle of fail-safe defaults states that, unless a subject is given explicit access to an object, it should be denied access to that object.

This principle requires that the default access to an object is none. Whenever access, privileges, or some security-related attribute is not explicitly granted, it should be denied. Moreover, if the subject is unable to complete its action or task, it should undo those changes it made in the security state of the system before it terminates. This way, even if the program fails, the system is still safe.

2.2.3 Economy of Mechanism

This principle simplifies the design and implementation of security mechanisms.

The principle of economy of mechanism states that security mechanisms should be as simple as possible.

If a design and implementation are simple, fewer possibilities exist for errors. The checking and testing process is less complex, because fewer components and cases need to be tested. Complex mechanisms often make assumptions about the system and environment in which they run. If these assumptions are incorrect, security problems may result.

2.2.4 Complete Mediation

This principle restricts the caching of information, which often leads to simpler implementations of mechanisms.

The principle of complete mediation requires that all accesses to objects be checked to ensure that they are allowed.

Whenever a subject attempts to access an object, the system should mediate the action. First, it determines if the subject is allowed to access the object. If so, it provides the resources for the required access. If the subject tries to access the object again, the system should check that the subject is still allowed to access the object. Most systems would not make the second check. They would cache the results of the first check and base the second access on the cached results.

2.2.5 Open Design

This principle suggests that complexity does not add security.

The principle of open design states that the security of a mechanism should not depend on the secrecy of its design or implementation.

The design should not be secret. The mechanisms should not depend on the ignorance of potential attackers, but rather on the possession of specific, more easily protected, keys or passwords. This decoupling of protection mechanisms from protection keys permits the mechanisms to be examined by many reviewers without concern that the review may itself compromise the safeguards. In addition, any skeptical user may be allowed to convince himself that the system he is about to use is adequate for his purpose. Finally, it is simply not realistic to attempt to maintain secrecy for any system which receives wide distribution.

2.2.6 Separation of Privilege

This principle is restrictive because it limits access to system entities.

The principle of separation of privilege states that a system should not grant permission based on a single condition.

Where feasible, a protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key. The relevance of this observation to computer systems was pointed out by R. Needham in 1973. The reason is that, once the mechanism is locked, the two keys can be physically separated and distinct programs, organizations, or individuals made responsible for them. From then on, no single accident, deception, or breach of trust is sufficient to compromise the protected information. This principle is often used in bank safe-deposit boxes. It is also at work in the defense system that fires a nuclear weapon only if two different people both give the correct command. In a computer system, separated keys apply to any situation in which two or more conditions must be met before access should be permitted.

2.2.7 Least Common Mechanism

This principle is restrictive because it limits sharing.

The principle of least common mechanism states that mechanisms used to access resources should not be shared.

Minimize the amount of mechanism common to more than one user and depended on by all users. Every shared mechanism (especially one involving shared variables) represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security. Further, any mechanism serving all users must be certified to the satisfaction of every user, a job presumably harder than satisfying only one or a few users.

2.2.8 Psychological Acceptability

This principle recognizes the human element in computer security.

The principle of psychological acceptability states that security mechanisms should not make the resource more difficult to access than if the security mechanisms were not present.

It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly. Also, to the extent that the user's mental image of his protection goals matches the mechanisms he must use, mistakes will be minimized. If he must translate his image of his protection needs into a radically different specification language, he will make errors.

2.3 Access control concepts

An access control system regulates the operations that can be executed on data and resources to be protected. Its goal is to control operations executed by subjects in order to prevent actions that could damage or steal data and resources.

2.3.1 Authorization versus authentication

Authorization and authentication are fundamental to access control. They are distinct concepts but often confused.

Authentication is the process of determining that a user's claimed identity is legitimate. It is based on one or more of the following factors:

- something you know (password, personal identification number – PIN, lock combination etc)
- something you have (smart card, automatic teller machine (ATM) card or key)
- something you are or a physical characteristic (fingerprint, retinal pattern, facial characteristic etc)

Authentication is normally stronger if two or more factors are used. A password can be guessed, a key lost, and face-recognition systems have a significant false positive rate, so using only one of these authentication methods may not provide an acceptable level of security. This is why banks require both cards and PINs to access ATMs rather than only a password, or only a key card. If the card were lost, a thief would have to guess the PIN in only three tries.

While authentication is the process of determining who you are, **authorization** determines what you are allowed to do. Authorization refers to a yes or no decision as to whether a user is granted access to a system resource. An information system must maintain some relationship between user IDs and system resources. This can be done either by attaching a list of authorized users to resources, or by storing a list of accessible resources with each user ID. Note that authorization necessarily depends on proper authentication. If the system cannot be certain of a user's identity, there is no valid way of determining if the user should be granted access.

2.3.2 Users, subjects, objects, operations and permissions

Almost any access control model can be stated formally using the notions of users, subjects, objects, operations and permissions and relationship between these entities. It is important to

understand these terms because they are used in the literature on access control and computer security.

The term **user** refers to people who interface with the computer system. In many designs, it is possible for a single user to have multiple login IDs, and these IDs may be simultaneously active. Authentication mechanisms make it possible to match the multiple IDs to a single human user.

An instance of a user's dialog with a system is called a **session**.

A computer process acting on behalf of a user is referred to as a **subject**. A user may have multiple subjects in operation, even if the user has only one login and one session.

An **object** can be any resource accessible on a computer system (e.g. files, peripherals, databases, and fine-grained entities such as individual files in a database records). Objects are traditionally viewed as passive entities that contain or receive information, although even early access control models included the possibility of treating programs, printers or other active entities as objects.

An **operation** is an active process invoked by a subject. Early access control models that were concerned strictly with information flow (i.e. read-and-write access) applied the term subject to all active processes, but RBAC models require a distinction between subject and operation.

Permissions (or privileges) are authorizations to perform some action on the system. It is a combination of object and operation. A particular operation used on two different objects represents two distinct permissions, and similarly, two different operations applied to a single object represent two distinct permissions.

2.3.3 Policy, models and mechanisms

While authentication mechanisms ensure that system users are who they claim to be, these mechanisms say nothing about what operations users should or should not perform within the system. In order to protect the system in such situations, it is necessary to use access control.

Access control is concerned with determining the allowed activities of legitimate users, mediating every attempt by a user to access a resource in the system. A given IT infrastructure can implement access control systems in many places and at different levels. Operating systems use access control to protect files and directories. Database management systems apply access control to regulate access to tables and views.

When considering any access control system, one considers three abstractions of control: *access control policies*, *access control models* and *access control mechanisms*.

Policies are high level requirements that specify how access is managed and who, under what circumstances, may access what information. Policies may pertain to resource usage within or across organizational units or may be based on need-to-know, competence, authority, obligation or conflict-of-interest factors. Furthermore, access control policies are *dynamic* in nature, in that they are likely to change over time in reflection of ever evolving business factors, government regulations and environmental conditions. However, because policy requirements can rarely be completely determined in advance, access control systems are best designed to flexibly accommodate a wide variety of changing policies.

At a high level, access control policies are enforced through a **mechanism** that translates a user's access request often in terms of a simple table lookup to grant or deny access. In general, access control mechanisms require that *security attributes* be kept about users and resources. User security attributes consist of things like user identifiers, groups and roles to which users belong, or they can include security labels reflecting the level of trust assigned to the user. Resource attributes can take on a variety of forms such as sensitivity labels, types or access control lists. In determining the user's ability to perform operations on resources, access control mechanisms compare the user's

security attributes to those of the resource. Other characteristics of access control mechanisms include *attribute review* and *management capabilities*. For example, can the access control system determine the permissions that are associated with a user or the users that can access a resource? Who can specify permissions? Can permission specification be delegated?

Rather than attempting to evaluate and analyze access control systems exclusively at the mechanism level, security models are usually written to describe the security properties of an access control system. Access control **models** are written at a level of abstraction to accommodate a wide variety of implementation choices and computing environments, while providing a conceptual framework for reasoning about the policy they support. Access control models bridge the rather wide gap in abstraction between policy and mechanism. Users see an access control model as an unambiguous and precise expression of requirements. Vendors and system developers see access control models as design and implementation requirements.

2.3.4 Reference monitor

The reference monitor (Figure 5) is an abstract concept whereby all accesses that subjects make to objects are authorized based on the information contained in an access control database. Conceptually, the reference monitor represents the hardware and software portion of an operating system that is responsible for the enforcement of the security policy of the system. The access control database is the embodiment of this policy in terms of subject and object attributes and access rights. When a subject requests access to an object, the reference monitor must perform a check, comparing the attributes of the subject with that of the object. Moreover, the reference monitor, with respect to some security policy, must control the specific checks that are made and all modifications to the access control database.

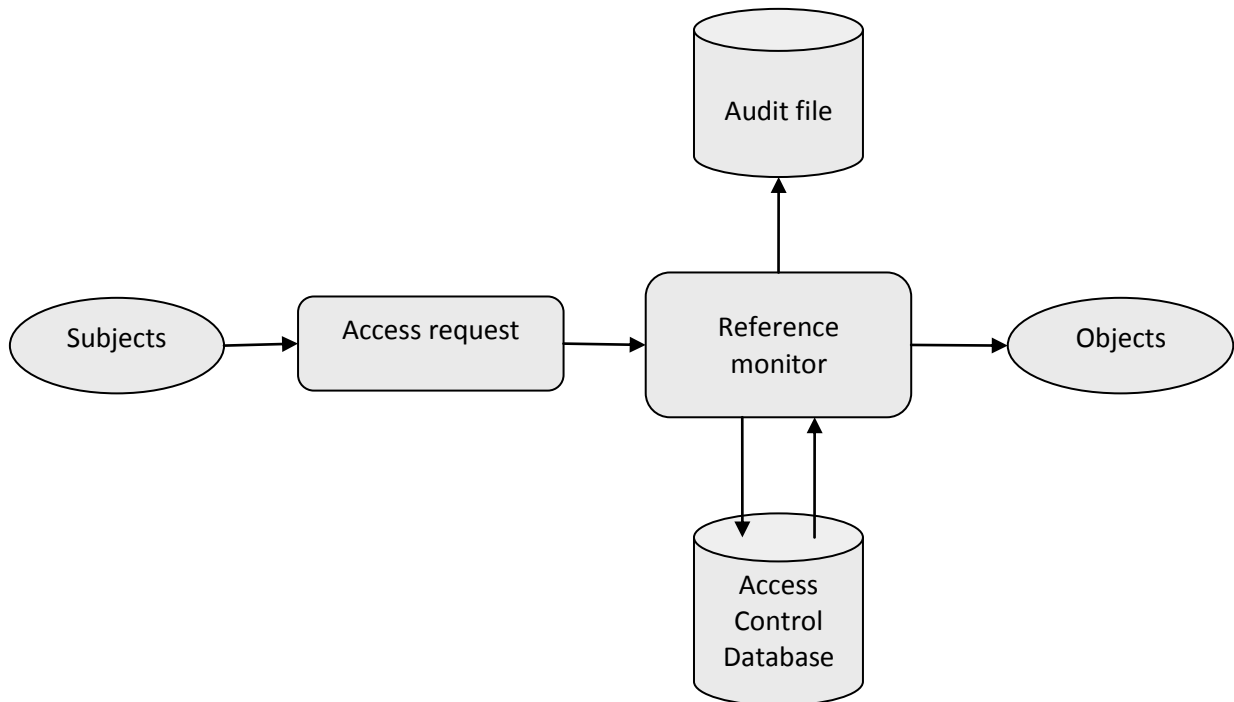


Figure 5 Reference monitor

As an abstraction, the reference monitor does not dictate any specific policy to be enforced by the system, nor does it address any particular implementation. But it defines an assurance framework that has been used for over 3 decades in the design, development and implementation of highly secure IT systems.

The abstract requirements of a reference monitor are comprised of three fundamental implementation principles, described as follows:

- **Completeness:** It must be always invoked and impossible to bypass.
- **Isolation:** It must be tamper-proof.
- **Verifiability:** It must be shown to be properly implemented.

The degree to which a system complies with these design principles has served as a metric for measuring the level of confidence in the correctness of the system's security controls.

The **completeness** principle requires that a subject can reference an object by invoking the reference monitor. Although this principle may seem intuitive, few mainstream operating systems completely adhere to this principle. There are two issues that make the principle hard to be met. The first issue is what are considered to be the objects in the system. In general, objects are interpreted to be any entities that can store information such as files, directories, memory. But there are not so obvious places where information is stored like file names, segments, processors, and status and error messages. The completeness principle requires that all objects must be protected – not just the obvious ones. The second architectural challenge pertaining to the completeness principle is the prevention of access to objects through methods (documented or otherwise) other than through the invocation of the policy-preserving access checker. For example, a subject could bypass a file system and issue a read request directly to the physical location of a file on disk.

The **isolation** principle states that the access mediation function is tamper-proof. It must be impossible for a penetrator to attack the access mediation mechanism in a manner that affects the proper performance of access checks. Even though most resource management systems are designed to protect themselves against accidental and overt break-in attempts, meeting the absolute requirements of the isolation principle of the reference monitor usually requires a security architecture consisting of both hardware and software features.

The principle of **verifiability** is met through software engineering practices and design criteria (e.g. code inspection and positive and negative testing). In some extreme cases, formal mathematical modeling, formal specification and verification techniques can be applied to prove the correctness of implementation.

[17] state three additional design principles seen as critical components of any access control system:

- **Flexibility:** The system should be able to enforce the access control policies of the host enterprise.
- **Manageability:** The system should be intuitive and easy to manage.
- **Scalability:** The system's management and enforcement functions should scale to the number of users and the number of resources that are scattered across the computing platforms of the host enterprise.

2.4 Access control models

2.4.1 Discretionary Access Control

Discretionary Access Control (DAC) is a mean of restricting access to objects based on the identity of users and/or groups to which the object belongs. Controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (directly or indirectly) to any other subject. To provide this discretionary control, DAC mechanisms usually include a concept of object ownership, where the object's owner has control permission to grant access permission to the object for other subjects. This definition of DAC has its origins with the Trusted Computer System Evaluation Criteria (TCSEC) [18] and is rationalized based on the DoD's regulatory requirements for need-to-know access to classified or sensitive information: "[...] no

person may have access to classified or sensitive information unless [...] access is necessary for the performance of official duties.”

DAC mechanisms tend to be very flexible and are widely used in commercial and government sectors. Throughout the mid 1980s and 1990s, virtually every computer vendor demonstrated DAC compliance by undergoing a C2 TCSEC (discretionary protection) evaluation.

Even though DAC mechanisms are in wide commercial use today, they are known to be inherently weak for two reasons: first, granting read access is transitive. For example, when Alice grants Bob read access to a file, nothing stops Bob from copying the contents of Alice’s file to an object that Bob controls. Bob may now grant any other user access to the copy of Alice’s file. Second, DAC mechanisms are vulnerable to “Trojan horse” attacks. Because programs inherit the identity of the invoking user, Bob may, for example, write a program for Alice that, on the surface, performs some useful function, while at the same time reads the contents of Alice’s file and writes the contents of the files to a location that is accessible by both Alice and Bob. Bob may then move the contents of the files to a location not accessible to Alice. Note that Bob’s Trojan horse program could have destroyed the contents of Alice’s file. When investigating the problem, the audit file would indicate that Alice destroyed her own file.

2.4.1.1 Access Control Lists

By far the most common mechanism for implementing DAC policies is through the use of Access Control Lists (ACL). When using ACLs, every piece of data, database, or application has a list of users associated with it who are allowed access. In this system, it is very easy for the security administrator to see which users have access to which data and applications. Changing access to the piece of information is straightforward; an administrator simply adds or deletes a user from the ACL.

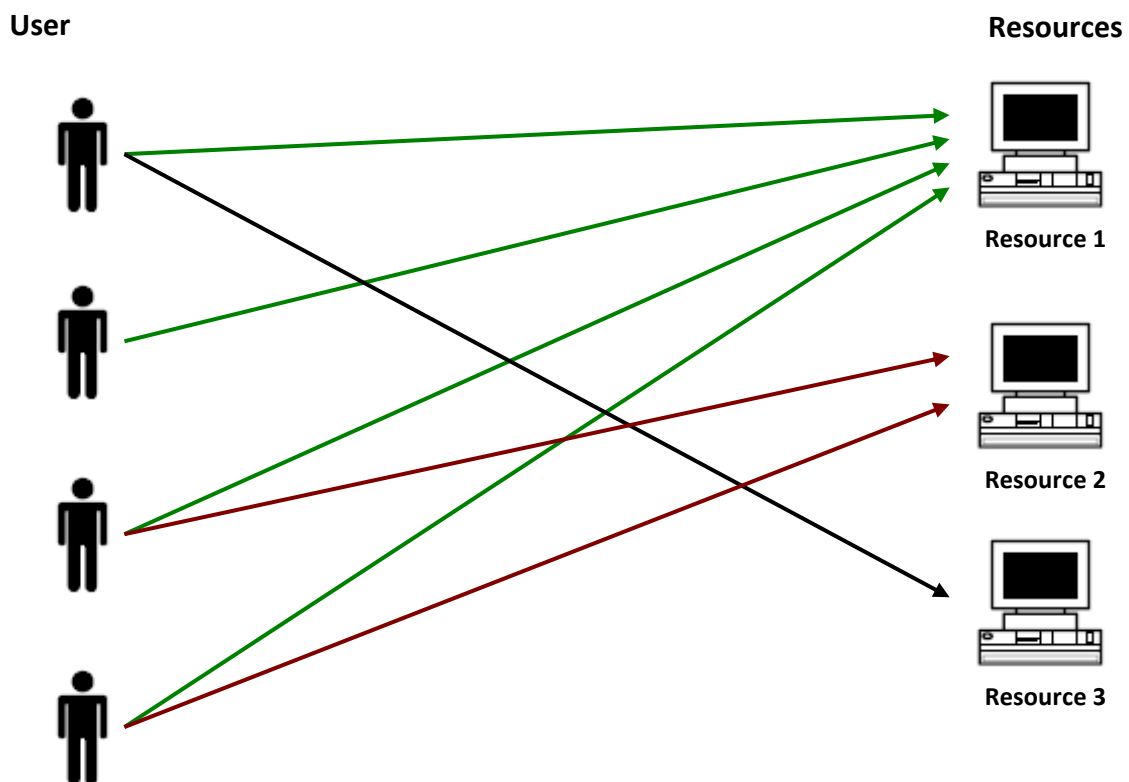


Figure 6 Discretionary Access Control using Access Control Lists

Each set of data or application has its own ACL, but there may or may not be a corresponding list that gives the administrator information on all of the pieces of information to which a particular user has access. Only by examining each piece of data individually and checking for access can the security administrator find any potential security violations. If all accesses by a particular user need to be revoked, the administrator must examine each ACL, one by one, and remove the user from each list.

When a user takes on different responsibilities within the organization, the problem gets worse. Rather than simply eliminating the user from every ACL, the network administrator must determine which permissions need to be eliminated, left in place, or altered. Administrators have made several attempts to improve ACLs. In some cases, users can be put into groups, making it easier to change the ACL. In other cases, elaborate rules can be applied to ACLs to limit access to particular pieces of data.

2.4.2 Mandatory Access Control

In addition to DAC policies, the TCSEC [18] defines Mandatory Access Control (MAC) policies that are known to prevent the Trojan horse problem. With regard to this policy, security levels are assigned to users, with subjects acting on behalf of users and objects. Security levels have a hierarchical and nonhierarchical component. For instance, the hierarchical components might include *unclassified* (U), *confidential* (C), *secret* (S), and *top-secret* (TS) while the nonhierarchical components may include NATO and NUCLEAR.

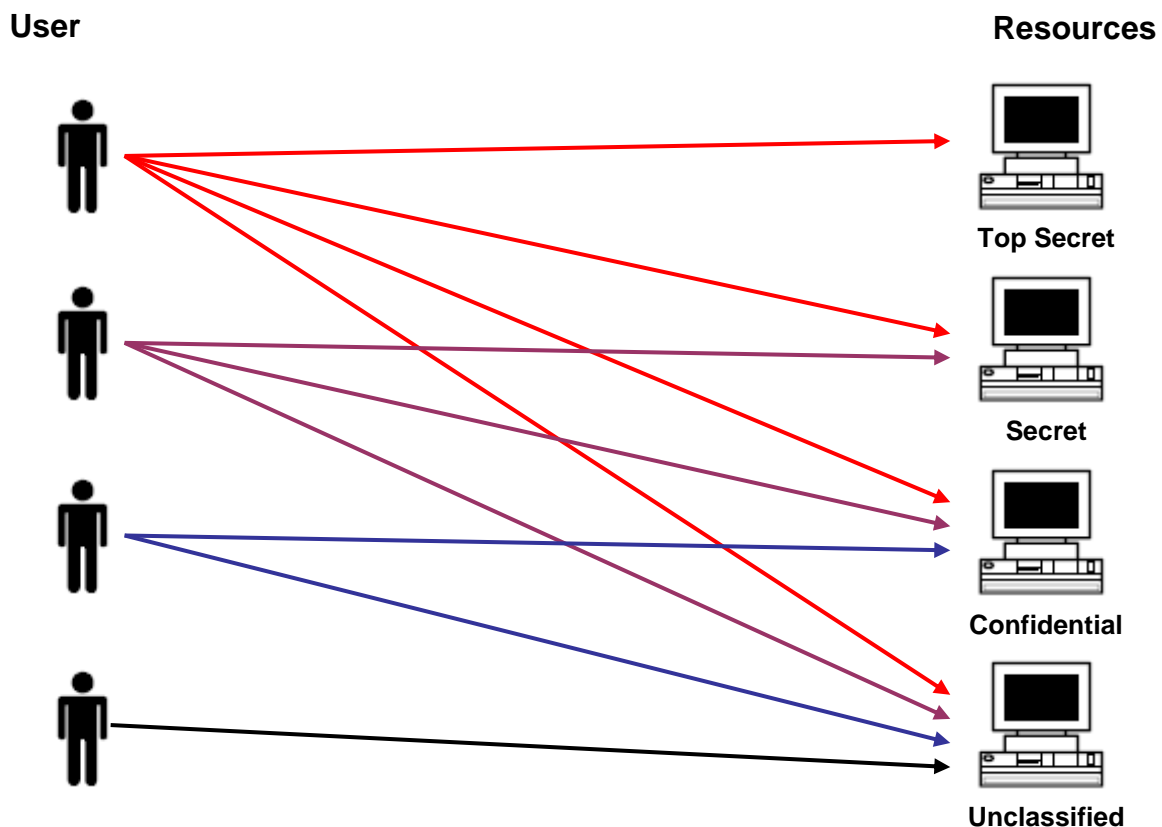


Figure 7 Mandatory Access Control

The security levels are partially ordered under a dominance relation, often written as “ \geq ”. For example, $TS \geq S \geq C \geq U$ and $S(\text{NATO}, \text{NUCLEAR}) \geq S(\text{NUCLEAR}) \geq S$. The security level of the user, often referred as the user’s *clearance* level, reflects the level of trust bestowed to the user and must always dominate the security levels that are assigned to the user’s subjects. For example, Chris who

is cleared to the S(NUCLEAR) level may initiate sessions at the S(NUCLEAR), S, C, or U levels. The security levels that are assigned to objects reflect the sensitivity of the contents of the objects.

2.4.2.1 Bell-Lapadula model

With respect to the security level of a subject and the security level of an object, the Bell-LaPadula model defines access control decisions in accordance with two properties:

- Simple security property: A subject is permitted read access to an object if the subject's security level dominates the security level of the object.
- Star property: A subject is permitted write access to an object if the object's security level dominates the security level of the subject.

Satisfaction of these properties prevents users from being able to read information that dominates (i.e. is above) their clearance level. The simple security property directly supports this policy, never allowing a subject to read information that dominates the invoking user's clearance level. The start property supports the MAC policy indirectly, by disallowing subjects from writing information of level x into a container (contents of an object) that could be subsequently read by a subject with a security level that is dominated by x. Intuitively, the star property prevents high information from ending up in a low container where a low user could read it.

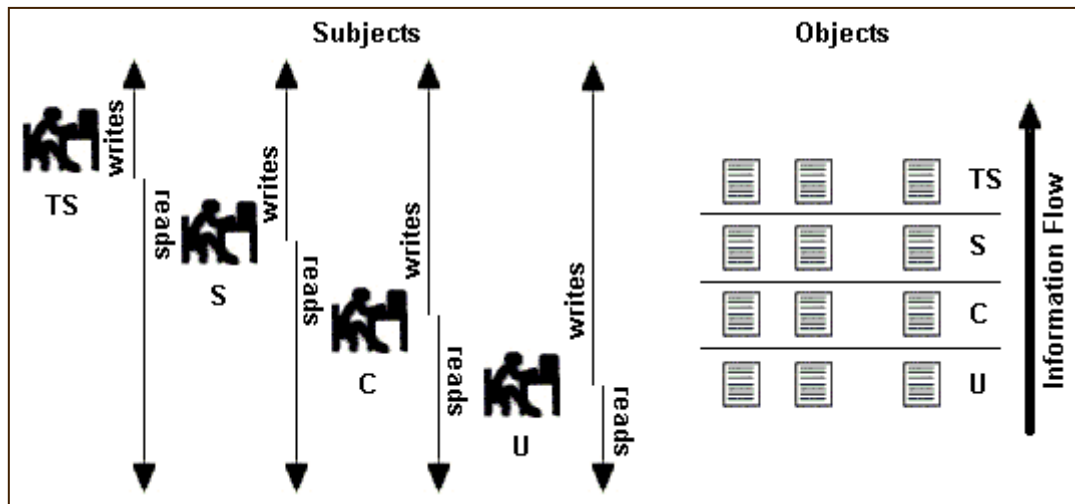


Figure 8 Hierarchical components in MAC.

2.4.3 Role Based Access Control

2.4.3.1 Brief History

The concept of Role Based Access Control (RBAC) began with multi-user and multi-application on-line systems pioneered in the 1970s. Though, only late in 1992, Ferraiolo and Kuhn paper called "Role-Based Access Control" defines RBAC model, with access permitted only through roles. Also, role hierarchies and constraints including separation of duty are formally defined.

Distributed Trusted Operating System (DTOS) based RBAC prototype was developed in 1994. DTOS project was a joint effort by the National Security Agency (NSA) and Secure Computing Corporation (SCC) to encourage strong, flexible security controls in next generation operating systems. DTOS was a successor to the Distributed Trusted Mach (DTMach) program. The NSA developed security enhancements for the process management, file system, and network protocol implementations in the Lites Unix single server. DTOS was part of a broad operating system security research program by the NSA known as Synergy. The Synergy program is no longer active, although the Flask and Security-Enhanced Linux projects have continued to pursue its goals. First patent

application in RBAC area was filed by IBM in 1994 in Europe. In the same year, Nyanchama and Osborn paper defined role graph model.

The formal model has been extended with various forms of separation of duty by Ferraiolo, Cugini, Kuhn in 1995. Next year, Sandhu, Coyne, Feinstein, Youman paper defines a family of RBAC models known as RBAC0, RBAC1, RBAC2 and RBAC3. As an attempt at rigorously defining RBAC features, a number of RBAC models have been proposed and implemented since then. These models had been independently proposed without any attempt at standardizing the RBAC features. The first step in the direction to standardization was done by Sandhu, Ferraiolo and Kuhn in 2000 [19]. They define consolidated RBAC model for proposed industry standard which has been adopted by the American National Standards Institute, International Committee for Information Technology Standards (ANSI/INCITS) in 2003 as an industry consensus standard INCITS 359:2004.

Following paragraphs will present the RBAC model as proposed to the industry standard.

2.4.3.2 Model overview

RBAC provides a valuable level of abstraction to promote security administration at a business enterprise level rather than at the user identity level. The basic role concept is simple: establish permissions based on the functional roles in the enterprise, and then appropriately assign users to a role or set of roles (Figure 9). With RBAC, access decisions are based on the roles individual users have as part of an enterprise. Roles could represent the tasks, responsibilities, and qualifications associated with an enterprise. Because the roles within an enterprise are relatively persistent with respect to user turnover and task re-assignment, RBAC provides a powerful mechanism for reducing the complexity, cost and potential for error in assigning user permissions within the enterprise. Because roles within an enterprise typically have overlapping permissions, RBAC models often include features to establish role hierarchies, where a given role can include all permissions of another role.

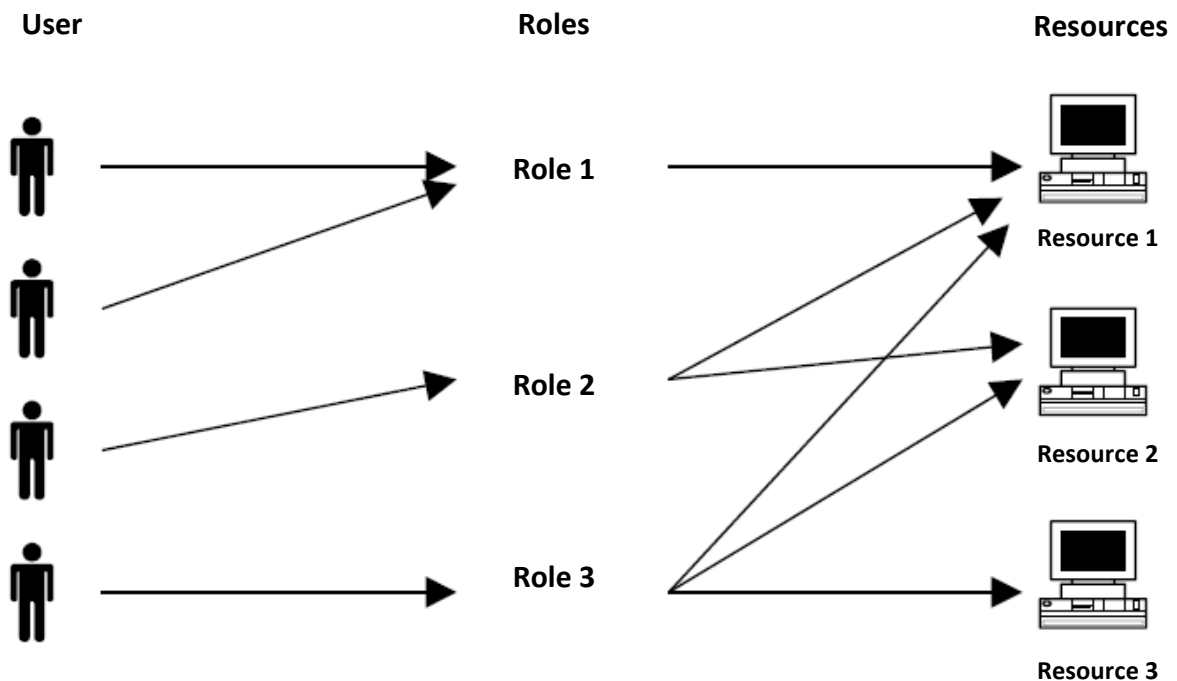


Figure 9 Role Based Access Control

RBAC is a rich and open-ended concept which ranges from very simple at one extreme to fairly complex and sophisticated at the other. It has been recognized that a single definitive model for RBAC is therefore unrealistic. Such a model would either include or exclude too much, and would

represent one point along a spectrum of choices. The NIST (National Institute of Standards and Technology) RBAC model is consequently organized in a four step sequence of increasing functional capabilities given below. These levels are cumulative in that each includes the requirements of the previous ones in the sequence: *Flat RBAC*, *Hierarchical RBAC*, *Constrained RBAC* and *Symmetric RBAC* (see Table 1).

Level	Name	RBAC Functional Capabilities
1	Flat RBAC (Figure 10)	<ul style="list-style-type: none"> - users acquire permissions through roles - must support many-to-many user-role assignment - must support many-to-many permission-role assignment - must support user-role assignment review - users can use permissions of multiple roles simultaneously
2	Hierarchical RBAC (Figure 11)	Flat RBAC + <ul style="list-style-type: none"> - must support role hierarchy (partial order) - level 2a requires support for arbitrary hierarchies - level 2b denotes support for limited hierarchies
3	Constrained RBAC (Figure 14, Figure 15)	Hierarchical RBAC + <ul style="list-style-type: none"> - must enforce separation of duties (SOD) - level 3a requires support for arbitrary hierarchies - level 3b denotes support for limited hierarchies
4	Symmetric RBAC (Figure 16, Figure 17)	Constrained RBAC + <ul style="list-style-type: none"> - must support permission-role review with performance effectively comparable to user-role review - level 4a requires support for arbitrary hierarchies - level 4b denotes support for limited hierarchies

Table 1 RBAC variations organized as levels.

2.4.3.3 Flat RBAC

Flat RBAC is illustrated in Figure 10. The features required of flat RBAC are obligatory for any form of RBAC and are almost obvious. The main issue with flat RBAC is the features that have been excluded.

Flat RBAC captures the features of traditional *group-based access control* as implemented in operating systems through the current generation. The NIST RBAC model recognizes traditional group-based access control as the first level of RBAC because it is widely deployed and familiar technology that serves well as the starting point for RBAC. At the same time by allowing additional more sophisticated levels of RBAC, the NIST model recognizes that RBAC is more than just another name for traditional but robust group-based access controls.

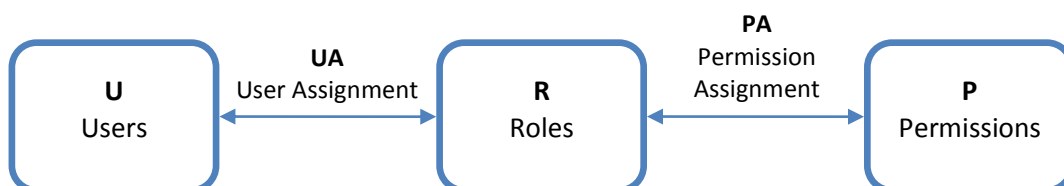


Figure 10 Flat RBAC.

The requirement that users acquire permissions through roles is the essence of RBAC. Flat RBAC does not exclude other means by which users can acquire permissions such as by direct assignment to the user or by means of security labels in lattice based access control.

Figure 10 shows three sets of entities called users (**U**), roles (**R**), and permissions (**P**). A *user* in this model is a human being or other autonomous agent such as a process or a computer. A *role* is a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role. A *permission* is an approval of a particular mode of access to one or more objects in the system. The terms authorization, access right and privilege are also used in the literature to denote a permission. Permissions are always *positive* and confer the ability to the holder of the permission to perform some action(s) in the system. Still, flat RBAC does not rule out the use of so-called negative permissions which deny access. The nature of a permission depends greatly on the implementation details of a system and the kind of system that it is. A general model for access control must therefore treat permissions as uninterpreted symbols to some extent. The exact nature of permissions in a system is left open by flat RBAC.

Flat RBAC requires that user-role assignment (**UA**) and permission-role assignment (**PA**) are *many-to-many* relations. This is an essential aspect of RBAC. The concept of a session is not explicitly a part of flat RBAC. A session corresponds to a particular occasion when a user signs on to the system to carry out some activity. The NIST model does not require support for sessions with discretionary role activation. It does require the ability to activate multiple roles simultaneously and in a single session.

Flat RBAC requires support for *user-role review* whereby it can be efficiently determined which roles a given user belongs to and which users a given role is assigned to. Permission-role review enables efficient answers to questions about which permissions are assigned to a role and which roles a permission is assigned to. In the NIST model requirement for permission-role review is deferred until level 4 in recognition of its intrinsic difficulty in large-scale distributed systems.

The flat RBAC model leaves open many important issues of RBAC that must be addressed in the implementation:

- scalability: there are no scalability requirements on the numbers of roles, users, permissions etc that should be supported
- nature of permissions
- support for discretionary role activation
- revocation behavior: revocation can occur when a user is removed from a role or a permission is removed from a role. How quickly the revocation actually takes place, particularly with respect to activity which is already under way, is left unspecified.
- role administration: who gets to assign users to roles and permissions to roles.

2.4.3.4 Hierarchical RBAC

Hierarchical RBAC is illustrated in Figure 11. It differs from Figure 10 only in introduction of the *role hierarchy* relation **RH**. Role hierarchies are often included whenever roles are discussed. They are also commonly implemented in systems that provide roles.

Role hierarchies are a natural means for structuring roles to reflect an organization's lines of authority and responsibility. Mathematically, these hierarchies are *partial orders*. A partial order is a *reflexive*, *transitive* and *anti-symmetric* relation. By convention more powerful (or *senior*) roles are shown toward the top of role-hierarchy diagrams, and less powerful (or *junior*) roles toward the bottom. There is strong consensus regarding the benefits of supporting arbitrary partial orders. Nevertheless there are products which support only limited hierarchies, but nevertheless provide substantially improved capabilities beyond a flat model. Hence, the recognition of two sub-levels in this context as follows:

- **General Hierarchical RBAC:** In this case there is support for an arbitrary partial order to serve as the role hierarchy.
- **Limited Hierarchical RBAC:** If any restriction is imposed on the structure of the role hierarchy then we are in this case. Most commonly, hierarchies are limited to simple structures such as trees or inverted trees.

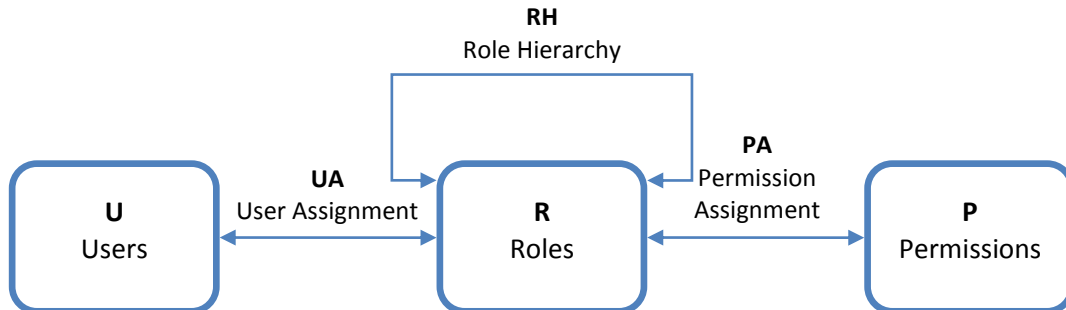
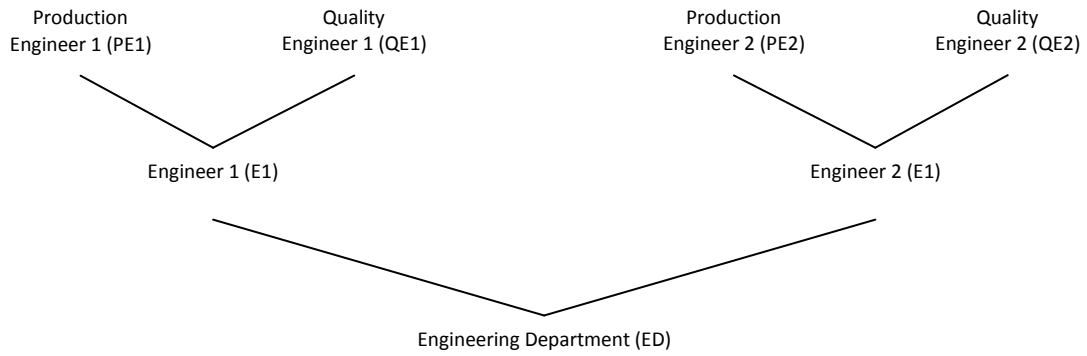


Figure 11 Hierarchical RBAC.

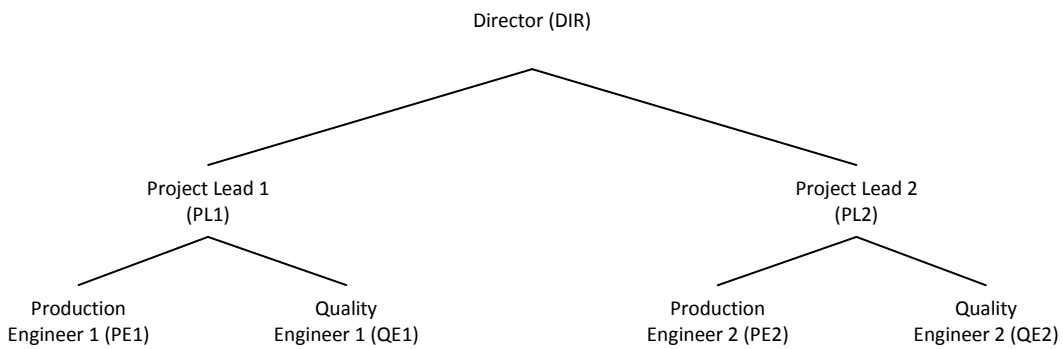
Figure 12 (a) shows an inverted tree hierarchy that might exist in a hypothetical engineering department. In these diagrams senior roles are shown towards the top with edges connecting them to junior roles. The inverted tree facilitates *sharing* of resources. Resources made available to the junior role are also available to senior roles. However, an inverted tree does not allow aggregation of resources from more than one role.

Figure 12 (b) shows a tree hierarchy in which senior roles aggregate the permissions of junior roles. Trees are good for *aggregation* but do not support sharing. In this hierarchy there can be no sharing of resources between the project 1 roles on the left and project 2 roles on the right.

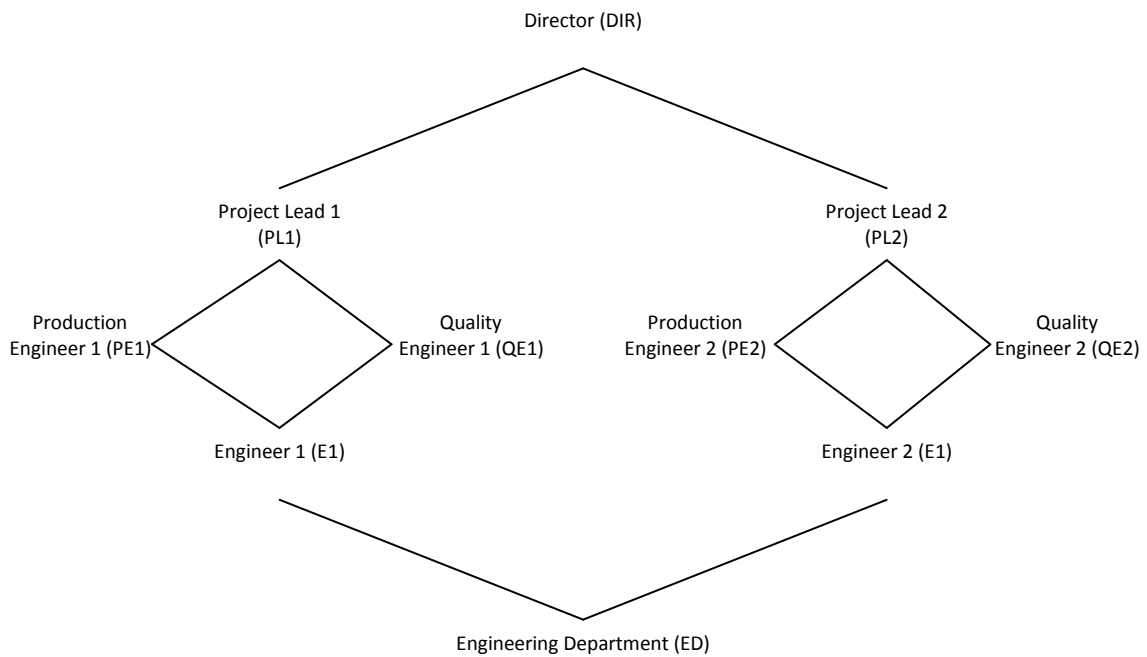
Figure 12 (c) shows a general hierarchy that facilitates both *sharing* and *aggregation*. This structure can be extended to dozens and even hundreds of projects within the engineering department. Moreover, each project could have a different structure for its roles. The example can also be extended to multiple departments with different structure and policies applied to each department. Practical hierarchies will typically have an irregular structure rather than the highly symmetrical construction of this example.



(a) A inverted tree hierarchy



(b) A tree hierarchy



(c) A general hierarchy

Figure 12 Example of role hierarchies.

We emphasize there is no requirement that there be a “senior most” role such as DIR in this example. Similarly, there is no requirement that there be a “junior most” role such as ED. The design of a suitable hierarchy is a matter of policy. The requirement is to support general hierarchies in level 2a and limited ones in level 2b.

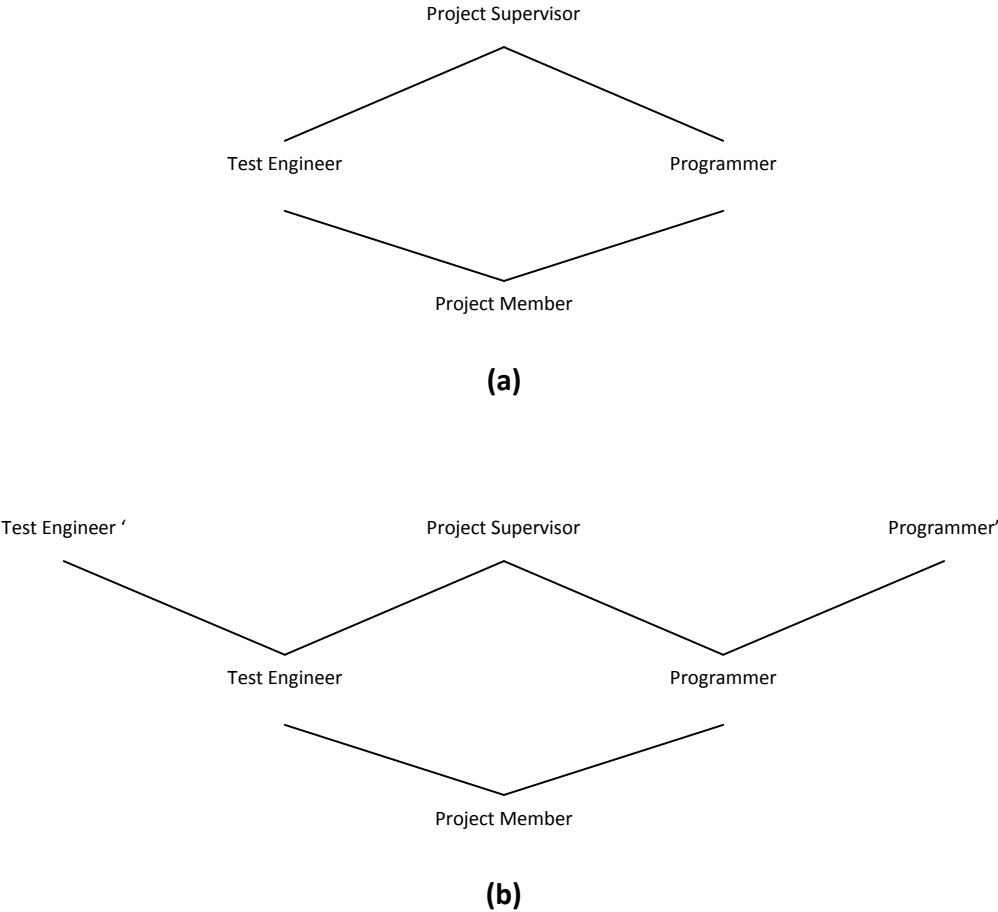


Figure 13 Example of limited inheritance.

Senior roles such as DIR in Figure 12 (c) are often considered dangerous because they aggregate too much power. It is possible to limit inheritance in role hierarchies as illustrated in the Figure 13. Figure 13 (a) shows that the Project Supervisor role inherits all permissions of the project. Figure 13 (b) on the other hand allows Test Engineers to have permissions in the Test Engineer' role that are not inherited by Project Supervisor. Test engineers are assigned to the Test Engineer' role whereas the Test Engineer role is simply a placeholder for those permissions of the Test Engineer' role that need to be inherited upwards. Roles such as Test Engineer' are called private roles.

There are two distinct interpretations of a role hierarchy that, have been discussed in the literature.

In one interpretation members of a senior role in the hierarchy are regarded as inheriting permissions from juniors. This is called the **permission-inheritance interpretation** and the hierarchy is called an **inheritance hierarchy**. Interpreting Figure 12 (c) as an inheritance hierarchy, when role PL1 is activated the permissions assigned to PL1, PEI, QEI, EI, ED and E are all available for use.

In the alternate interpretation, activation of a senior role does not automatically activate permissions of junior roles. This is called the **activation interpretation** and the hierarchy is called an **activation hierarchy**. In this case activation of role PL1 does not activate permissions of the junior

roles. Each junior role must be *explicitly* activated to enable its permissions in a session. It is possible to have both interpretations simultaneously applied.

The NIST model leaves open the exact meaning of role hierarchies since multiple interpretations are possible.

2.4.3.5 Constrained RBAC

Constrained RBAC, shown in Figure 14 and Figure 15, adds constraints to the hierarchical RBAC model. Constraints may be associated with the user-role assignment (*static*, Figure 14), or with the activation of roles within user sessions (*dynamic*, Figure 15). Separation requirements are used to enforce conflict of interest policies that organizations may employ to prevent users from exceeding a reasonable level of authority for their positions.

Separation of duty refers to the partitioning of tasks and associated privileges among different roles so as to prevent a single user from garnering too much authority. The motivation is to ensure that fraud and major errors cannot occur without deliberate collusion of multiple users to this end. Within an RBAC system separation concepts are supported by the principle of *least privilege*.

The NIST model allows for both static and dynamic separation of duty, but leaves open which of these should be supported and exactly in what form.

Static Separation of Duties (SSD) specifies constraints on the *assignment of users to roles*. This means that if a user is authorized as a member of one role, the user is prohibited from being a member of a second role. The SSD policy can be centrally specified and then be uniformly imposed on specific roles.

Constraints are inherited within a role hierarchy. Because a containing role is effectively an instance of its contained roles, no SSD relationship can exist between them.

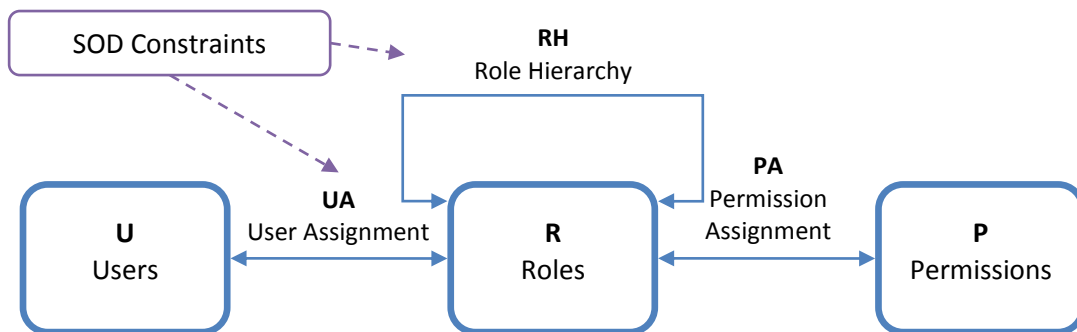


Figure 14 Constrained RBAC – Static SOD.

While SSD provides an organization with the capability to address potential conflict-of-interest issues at the time a user's membership is authorized for a role, with **Dynamic Separation of Duty (DSD)** it is permissible for a user to be authorized as a member of a set of roles which do not constitute a conflict of interest when acted in independently, but produce policy concerns when allowed to be acted in simultaneously. Although this effect could be achieved through the establishment of an SSD relationship, DSD relationships generally provide the enterprise with greater operational flexibility.

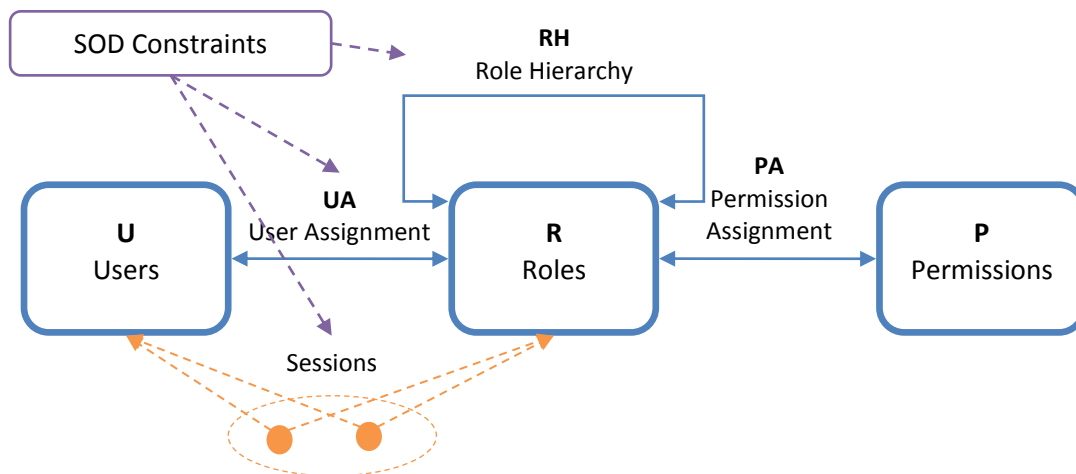


Figure 15 Constrained RBAC – Dynamic SOD.

Note that unlike roles in an SSD relation, roles in a DSD relation can be hierarchically related through the containment relation. This is consistent with the DSD property of restricting simultaneous activation of roles and that of a role hierarchy as a representation of a user's implicit and explicit authorizations for a role. As such, authorization and activation can be treated as independent notions.

The NIST model requires role hierarchies as a prerequisite in systems providing separation of duty because most of the benefits of RBAC are tightly integrated with the provision of hierarchies. Separation of duty mechanisms implemented without hierarchies have serious limitations on flexibility and functionality.

2.4.3.6 Symmetric RBAC

Starting at Level 1 RBAC systems are required to establish and maintain many-to-many relationships among user-role and permission-role assignments. Among these relations level 1 and level 2 RBAC systems require an interface for the review of user-role assignments. Level 1 requirements include the establishment of the set of roles that are directly assigned to a user. Level 2 RBAC extends coverage of user-role review to include not only the roles that are assigned to a user but also the roles that are inherited by the roles that are assigned to the user.

Level 4 RBAC further extends these requirements to include an interface for permission-role review with respect to a defined user or role. These requirements pertain to the type of data that is returned to the administrator as a result of a review, the ability to select direct or indirect permission assignments, and for distributed systems the ability to select the target systems in which the permission review will be applied.

Level 4 or symmetric RBAC requires that the *permission-role review* interface provide the capability to return any one of two types of results. These results include the complete set of objects that are associated with the permissions assigned to a particular user or role, or the complete set of operation and object pairs that are associated with the permissions that are assigned to a particular user or role. As an option on this query symmetric RBAC requirements further include the ability to selectively define direct and indirect permission assignment.

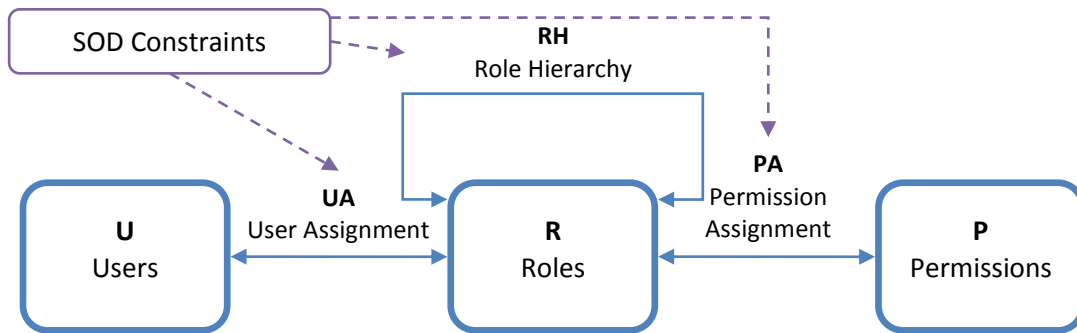


Figure 16 Symmetric RBAC – Static SOD.

Direct permission assignment pertains to the set of permissions that are assigned to the user and/or to the role(s) for which the user is assigned.

Indirect permission assignments pertain to the set of permissions that are included in the direct permission assignment in addition to the permissions that are assigned to the roles that are inherited by the roles assigned to the user.

As a further option on a permission query symmetric RBAC requires the ability to select the target systems for which the review will be conducted.

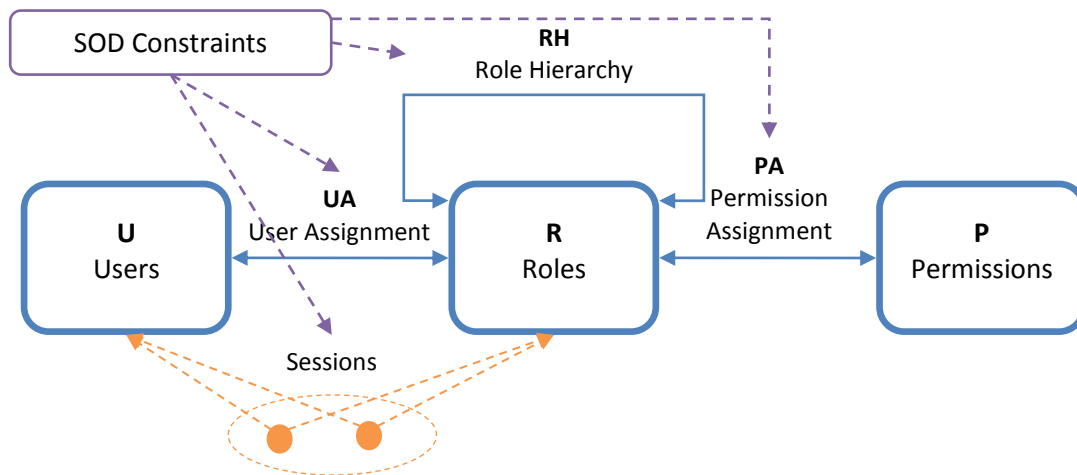


Figure 17 Symmetric RBAC – Dynamic SOD.

2.5 Conclusions

As organizations increase the functionality and information offered on internal and external networks, controlling access to information and other resources becomes more important and complex. Organizations must develop and enforce access policies that protect sensitive and confidential information; prevent conflict of interest; and protect the system and its contents from intentional and unintentional damage, theft, and unauthorized disclosure. Security failures can disrupt an organization’s operations and can have financial, legal, human safety, personal privacy, and public confidence impacts.

Safeguarding information resources can be very complicated and expensive. Network administrators must maintain access control lists that specify the resources each user is allowed to access. They must issue passwords and permissions to enforce the access lists and update them as personnel change and as users’ needs and permissions change. Maintaining access while enforcing a comprehensive, coherent security policy has become an expensive and complex task. Simplifying it could have an important impact on the cost and effectiveness of electronic resource access policies.

RBAC is a technology that offers an alternative to traditional DAC and MAC policies. RBAC allows companies to specify and enforce security policies that map naturally to the organization's structure. That is, the natural method for assigning access to information in a company is based on the individual's need for the information, which is a function of his job, or role, within the organization. RBAC allows a security administrator to use the natural structure of the organization to implement and enforce security policy. This technology decreases the cost of network administration while improving the enforcement of network security policies.

The following chapter describes the ATLAS TDAQ requirements from security point of view and draws the access control model needed to fulfill its needs.

Chapter 3

Access control solution for ATLAS Online computing system

3.1 Background and scope

The ATLAS experiment comprises a significant number of hardware devices, software applications and human personnel to supervise the experiment operation. Their protection against damages as a result of misuse and their optimized exploitation by avoiding the conflicting accesses to resources are key requirements for the successful running of ATLAS.

From the organization point of view, the ATLAS experiment's computing infrastructure is supported in CERN by the Information Technology (IT) department [20]. This relation imposes constraints on how ATLAS sets up its cluster and internal services because some of them must be integrated with IT services. On the other hand, CERN IT leverages the user management from the administrative and bureaucratic point of view; it also offers service for network monitoring, offline databases, emails, websites and many more.

The main security threat acknowledged by experiment management is **denial of service** because having beam in LHC and not benefiting of it due to downtime of the experiment acquisition software translates directly in waste of resources: human, hardware, financial and, last but not least, time! **Modification** threat can contribute to the denial of service: modification of experiment functionality by changing hardware or software configuration which, if done improperly, may lead to hardware damages and possible to human injuries.

The "attackers" for these threats are in fact the experiment's users themselves, being them physicists or engineers. The number of users accessing the experiment resources from CERN and external institutes is considerable. Additionally, the users are characterized by a high mobility and various levels of expertise (from students to experienced researchers). Hence, the risk of experiment resources misuse due to mistakes on part of the user or simply lack of information is estimated as very high.

The other threats mentioned in 2.1.1 are considered too in ATLAS, but they are addressed mainly by IT's services which, integrated in ATLAS, eliminate them or diminish their chances to occur.

The previous experiments run at CERN did not address all the security requirements (2.1.3) in a consistent manner and, especially, the **accountability** one. Researchers have used group accounts in their daily tasks: one user account known by all people in a group with the password shared in an informal way. The reasons for wanting to use group accounts comes from the natural splitting of users into categories of people and tasks which they are allowed to do, for example some users are shifters, detector experts, TDAQ experts. Given the large computing power put in place by the ATLAS experiment and the potential interest of worldwide attackers to exploit it, the security level in the experiment must be increased. Hence the accountability and traceability of actions became mandatory and it has been decided to have user based authentication and not group based authentication as it used to be.

In order to keep the categorization provided by group accounts and still have the user authentication/authorization for accountability and traceability, it has been decided in TDAQ to

implement the Role Base Access Control model which fulfills these needs, as outlined in 2.4.3, and brings also more flexibility in the access control management. In this way, the **psychological acceptability** (2.2.8) security principle is carefully addressed by making the transition to the new model smoother for the physicists and diminishing the resistance to change. However, concerns did exist and the physicists questioned the opportunity of such change in the way they work, being afraid that the research will be hindered by this new access control model. Table 2 summarizes most frequently asked questions and how they have been addressed.

Table 2 Users' frequently asked question on the access control in ATLAS experiment.

Question	Answer
<i>We are all part of ATLAS collaboration. Why do you want to check on us? We don't need access management (neither authentication nor authorization...)</i>	One of the primary goals of this mechanism is to reduce the risk of resources misuse. Given the high number of experiment resources, this risk is assessed as very high. Secondly, the high number of users increases also the risk of compromising computing accounts credentials. Hence the traceability and accountability are of paramount importance when dealing with security incidents.
<i>We are all experts. Why do you want to limit our freedom?</i>	The access control is meant to provide the user with the set of roles necessary for his or her tasks. This is also in line with the "Least Privilege" security principle. So the freedom is not limited because, in the end, is a matter of policies definition.
<i>What happens if access management doesn't work and experts cannot do what they need?</i>	The access management solution will have a turn off functionality to disable it in case of severe system malfunction. However, the changes to happen are minimized by designing the solution with high availability in mind.
<i>If an expert is asked to intervene, he cannot waste time asking for access.</i>	Experts will have necessary rights by default, but they still need to inform the shift leader about his intentions so that activities in the experiment environment are synchronized. The activation of expert roles is designed to be at a "click" distance.
<i>Will the shared folder disappear too?</i>	The need for shared folders modifiable by sets of people was never questioned. Moving from group accounts to individual accounts does impact the usage of shared folders. This is a matter of organizing the permissions on the files and folders.

3.1.1 Previous work

The access control issue in the ATLAS Trigger and Data Acquisition (TDAQ) system was addressed first time by an access management component [21] using the RBAC model. This implementation was limited to protection of the applications integrated in the TDAQ system that communicate over a CORBA (Common Object Request Broker Architecture) based inter process communication mechanism. The access management performed authorizations only while the TDAQ

system is running. No access control was provided for applications that run independently of the TDAQ or on tools accessible from Linux shell. Nevertheless, this first implementation confirmed that the RBAC model is the right one for ATLAS TDAQ needs.

The work described by this thesis extends the scope of access control to whole ATLAS Online computing cluster: from the remote access to the cluster to the protection of command line tools on the cluster nodes, login to the Control Room desktops and, of course, the data acquisition software itself. The solution we describe in the following chapter can be easily integrated in future with other applications or tools thanks to the Application Programming Interface libraries made available for this purpose.

3.2 High level design

The ATLAS experiment operational model defines activity areas with their tasks and responsibilities for various systems (e.g. detectors). The organizational structure resulting from this operational model is naturally reflected by the roles associated to systems. Each system in turn is organized internally in subsystems and the best mapping of this characteristic is the role hierarchy. Therefore, the Hierarchical RBAC model is the best approach for an access control schema in the ATLAS experiment. At the same time, the systems considered critical for the experiment operation may require a higher protection level by enforcing the SSD or DSD constraints.

At this stage, the ATLAS access control system implements the Hierarchical RBAC model (described in 2.4.3.4) and allows for extensions with SSD or DSD constraints.

3.2.1 Access Control System Architecture

The scope of the ATLAS access control is the ATLAS online computing system [22] where most of the hardware and software resources are located or are controlled from. This wide scope requires a design for the access control system that allows for a centralized management of the access policies and, at the same time, permits implementations from the lowest operating system level to the highest software application level.

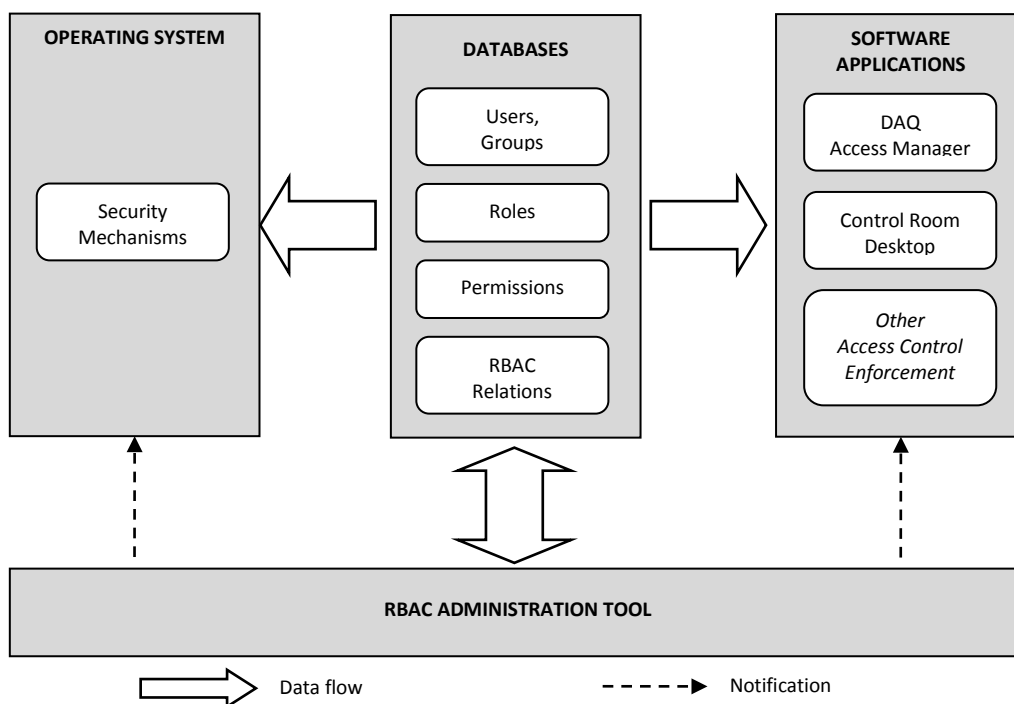


Figure 18 Access control software entities in the ATLAS online computing system.

Figure 18 provides an overview of the ATLAS access control software entities and their relations to meet these requirements. The following paragraphs detail the main software entities in the ATLAS online computing system involved in the implementation of the RBAC.

The *databases* are the coordination points for the access control within the experiment context. All the RBAC elements and relations can be defined in there: the users, the roles, the permissions, the roles hierarchies, the user assignments to role, and the permission assignment to role. The databases are the single point of storage for the RBAC constituents. There are multiple advantages of this approach: less data repositories to secure, less redundancy mechanisms to implement, and better control over the coherence of the access control policies among all the systems.

The access control policies are defined in the databases as RBAC relations between users, roles and permissions. The enforcement of these access control policies is the responsibility of the software entities interested in the protection of their functionality and the data they manipulate.

The *operating system* running on the cluster nodes is the software entity with the largest spectrum of functionalities. The cluster nodes play various roles in the online computing system from the service providers (e.g. file servers, boot servers, application gateways, network services) to processing nodes with functions in the data acquisition. The users' access to each of the cluster nodes is restricted with the operating system native access control mechanisms and, in addition, taking into account the role of the user in the system. The mechanisms are detailed later in the system administration enforcement example.

The *software applications* executed in the operating system environment may be able to control directly hardware devices in the ATLAS experiment infrastructure or to access data sources with configuration parameters for hardware and software systems. These types of software entities are also subject to access control restrictions and they are responsible for enforcing the policy in their working area.

While the databases are the passive entities in the access control implementation, the operating system and the software applications are active entities which enforce the access control policies.

The last software entity is the *RBAC Administration Tool* which has the most complex set of functionalities and it is presented in detail in a dedicated paragraph.

3.2.1.1 RBAC Administration

The RBAC Administration Tool is responsible for the management of most of the RBAC elements sets and relations. The functionalities of the RBAC Administration Tool are split over three categories corresponding to the following components: Administrative Component, Users Sessions Component, and Review Component. There are two categories of users for this tool: the administrators and the shifters. The administrative component is controlled by the administrators, the users sessions component by the shifters and the review component by both user categories. The following paragraphs detail the RBAC Administration Tool components enumerated above.

3.2.1.1.1 Administrative Component

This component is in charge of the creation and maintenance of the RBAC elements sets and relations: create and delete roles, define resources and actions, define permissions by associating actions to resources, build role hierarchies, assign/revoke roles to/from users, assign/revoke permissions to/from roles, and allow to define SSD or DSD relations. The user accounts creation and deletion operations are not part of this component's functionalities and are detailed later in 4.3.2.

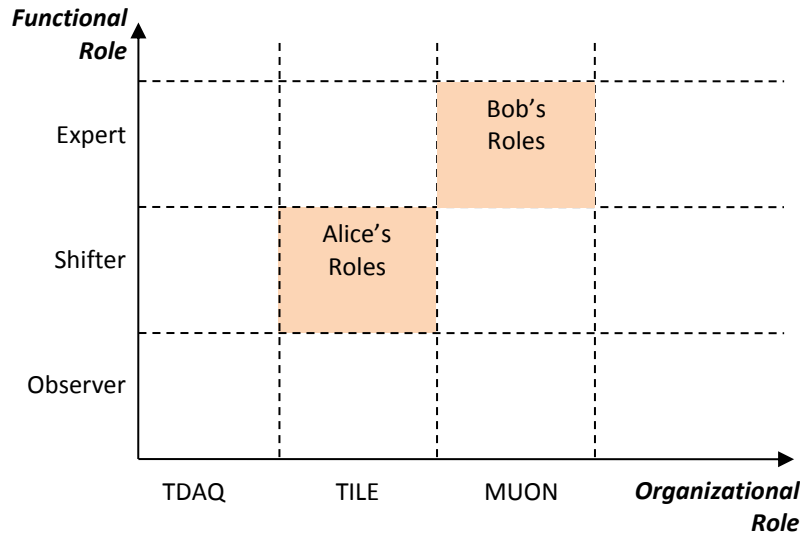


Figure 19 The organizational and functional roles

Figure 19 represents the two types of roles a user can have: the organizational roles and the functional roles. The organizational roles correspond to the ATLAS administrative groups or projects (e.g. subdetector group such as Hadronic Calorimeter (TILE), Electromagnetic Calorimeter (LAR), Muon, or TDAQ). The functional roles are associated to the expertise levels within an administrative group or project (e.g. administrator, expert, shifter, or observer). The set of roles **assigned** to a user comprises pairs of both types of roles in order to better describe the user's abilities and permissions in the ATLAS system. For example, the user Alice is a member of the TILE group and has been trained to be a shifter, so she's assigned to the role TILE-Shifter (inherits the organizational role TILE and the functional role shifter). The user Bob member of the MUON group has the highest level of expertise, so he is entitled with the MUON-Expert role (inherits the roles MUON and shifter). This kind of roles is assignable directly to users and is called from now on "**role assignable**".

There can be also **roles not assignable** to users which can be used in a role hierarchy for the sole purpose of aggregating permissions.

The role hierarchies incorporate the knowledge about the administrative group and project hierarchies in the ATLAS experiment. The hierarchies are defined for the functional roles: the shifter includes the observer's privileges, and the expert includes the shifter's privileges for example. The hierarchy granularity can be increased by defining roles per tasks within the same group or project. Here, the roles not assignable to users can facilitate this increased granularity.

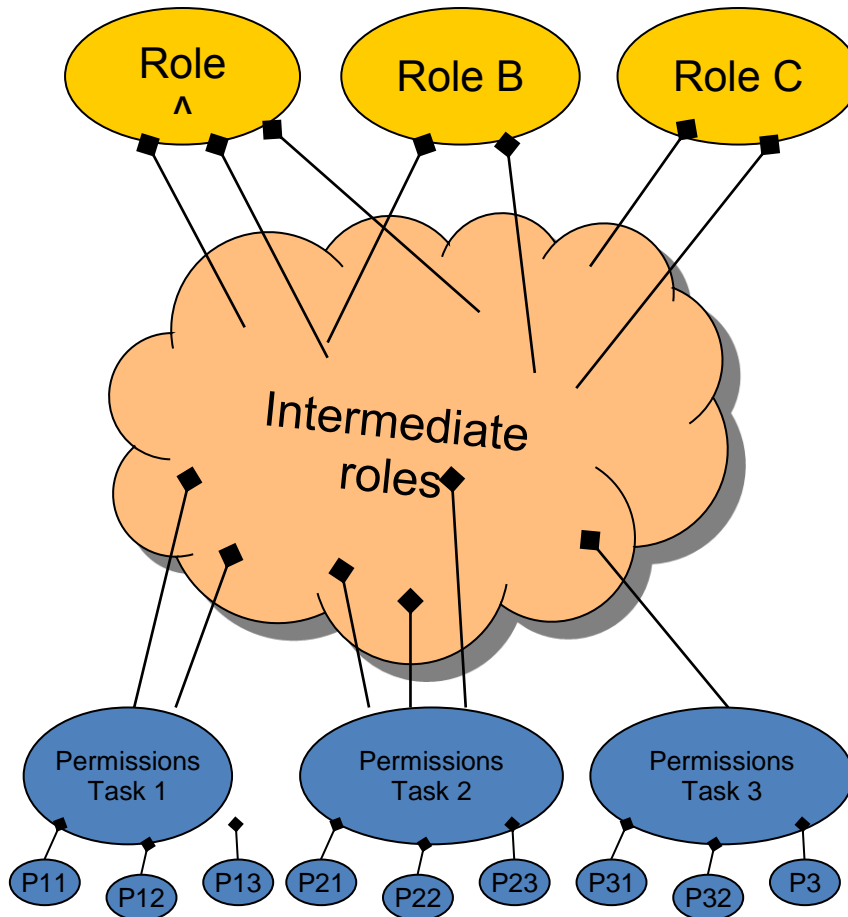


Figure 20 Task permissions aggregation into intermediate non assignable roles

Figure 20 shows how at the bottom of hierarchies the tasks permissions are gathered in groups which are assigned to intermediate not assignable roles, so that, in the end, the roles assignable to users (Role A, B, C) to inherit from intermediate roles the set of permissions necessary for each specific role. Sharing of permissions is of course possible by using the same intermediate role in two distinct top level roles.

In order to keep the administrative task as intuitive as possible, the roles hierarchies are limited to tree or inverted tree structures. The roles hierarchy example depicted in Figure 21 is composed of 2 types of roles hierarchies:

- *Tree hierarchy* (e.g. Shift Leader, TDAQ Shifter, DCS Shifter): senior roles aggregate the permissions of junior roles. Trees are good for aggregation but do not support sharing.
- *Inverted tree hierarchy* (e.g. Observer, TDAQ Shifter, DCS Shifter): senior roles are shown towards the top with edges connecting them to junior roles. The inverted tree facilitates sharing of resources. Resources made available to the junior role are also available to senior roles.

The resources types and the actions are predefined for each software entity designed to incorporate an access control mechanism. Therefore, the Administrative Component defines permissions in terms of what action is allowed on what resource value. For example, the predefined resource type “process manager” with the attribute “process name” has associated the predefined actions type “start” and “terminate” with the attributes “time”. A permission in this case could be: *Allow action type “start” with value “12:00” for attribute “time” to be performed on the resource type “process manager” with value “kdestart”.*

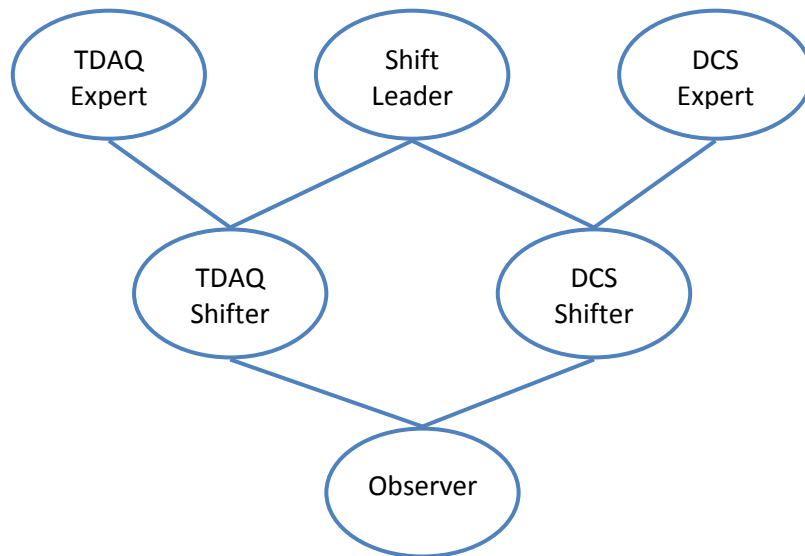


Figure 21 Example of roles hierarchy in ATLAS.

3.2.1.1.2 Users Sessions Component

The users in the ATLAS experiment have at most one session active at a time. The session creation and destruction is the responsibility of the Users Sessions Component. When a user session is created, a subset of roles is enabled from the set of roles assigned to the user. The enabling operation checks in addition the validity of the enabled subset of roles against the DSD relations if any were defined. The user session is visible in all the software entities with the access control implementations, and they are responsible for policy enforcement with the subset of roles enabled in the session.

3.2.1.1.3 Review Component

The administrator and shifter users may want to view the current status of the RBAC elements and relations. This goal is accomplished by the Review Component. It is able to display the permissions, the roles, the roles inheritance relationships, the user-to-role assignments, and the permission-to-role assignments.

The Administrative Component and the Users Sessions Component can change the RBAC elements and relations during the ATLAS experiment operation, and the new access policies must be enforced immediately everywhere. The access control implementations may cache some RBAC specific data to minimize the performance impact over the system they protect, so the cache is invalidated each time the RBAC Administration Tool notifies them about a change in the RBAC elements or relations.

The multitude of functions the RBAC Administration Tool is able to perform and their consequences on the good operation of the ATLAS experiment, makes from this tool a good candidate for an access control implementation. Indeed, the tool should protect itself with access control policies allowing only the administrators and shifters users to control it.

3.2.2 Access Control Enforcement

This section addresses the RBAC enforcement in the System Administration area, the Control Room Desktop implementations, and the TDAQ software.

3.2.2.1 System Administration

The access control in the System Administration concerns the protection of the software infrastructure in the ATLAS experiment and the operating system running on each online computing

node. Since more than 90% of the machines are running a Linux operating system (based on Scientific Linux CERN distribution [23]), the following chapters focuses on that operating system. The access control implementation on Windows OS is not in the scope of our work described in this thesis, but the integration with ATLAS RBAC is addressed. The few machines running the Windows OS are used by the ATLAS Detector Control Systems (DCS) where they run the PVSS SCADA software [24]. The access control is then implemented at the application level by the PVSS Access Control tool [25] extended on top of JCOP framework [26]. This extension was designed to meet CERN's requirements on access control [27]. This development has enabled the use of the RBAC specific data from the databases presented in the previous section to enforce the ATLAS access policies.

The software infrastructure [22] provides generic services like the network services (Domain Name System (DNS), Network Time Protocol (NTP), Dynamic Host Configuration Protocol (DHCP)), the network booting service for the online computing nodes (network booted for easy administration), the users information, and the users home directories.

The first protection level for the ATLAS experimental area (Figure 22) is provided by the Application Gateways which separate the ATLAS Technical Control Network (ATCN) from the rest of the CERN network. The users outside the experimental area and wishing to login to machines in the ATLAS Control Network have to first login to the Application Gateways and then hop to the desired machine. The Application Gateways are the single access point to the experimental area and it is therefore the best place to implement access.

Once the users have hopped through the Application Gateway, they are free to attempt to login to any machine in the experimental area (the online computing nodes, the file servers etc). However, each machine is protected with the Linux native security mechanisms extended to take into account the user's roles defined by ATLAS policy and the machine's functionality.

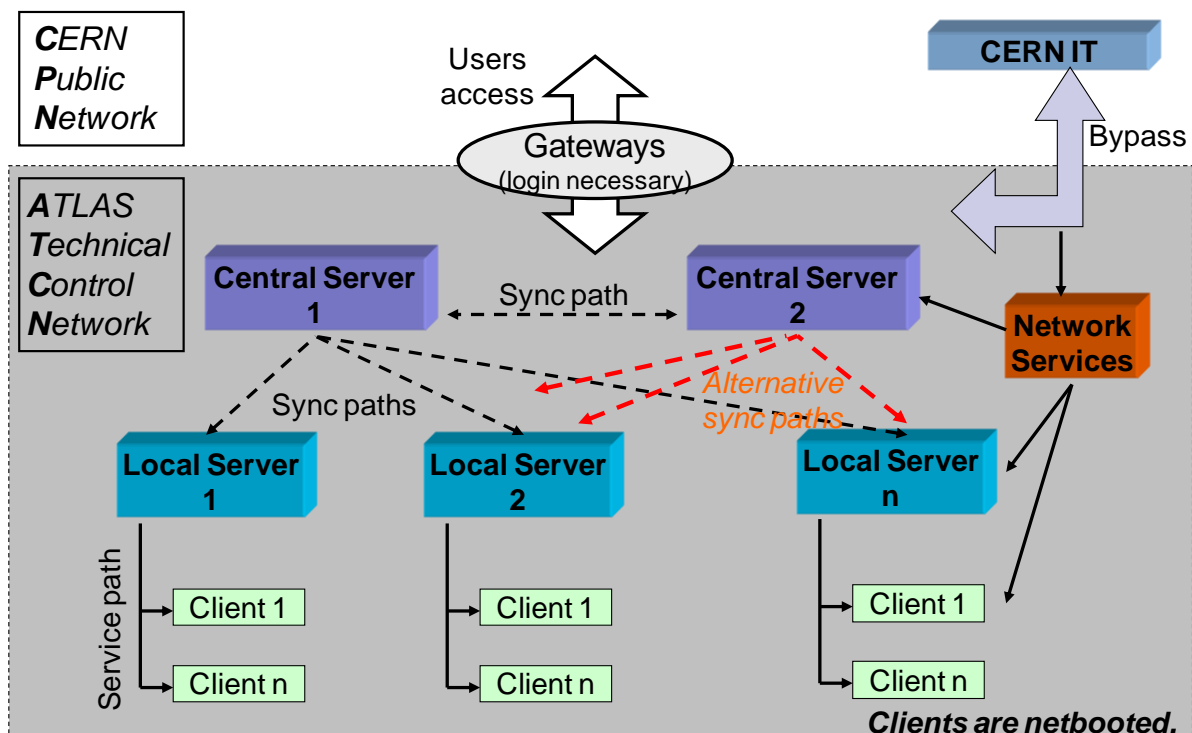


Figure 22 The ATLAS system architecture.

The computers functionalities in the ATLAS experimental area vary from public nodes where the users can display web pages of the safety information to the control room nodes where the detectors operational parameters can be changed. While the security level should be minimum for the first case (the users must be able to view the safety instructions as quickly as possible without

any role checking, but only the security officers must be able to edit and update these instructions), the second case needs the maximum security level (the user at the control room desktop console must have all the necessary roles enabled to be able to change the detector running conditions). This variety of computer functionalities requires a solution to configure the access control granularity for each individual node. This goal can be achieved by adding more access control filters with finer restrictions each time the security level needs to be increased for a node, so that the node's access control is in the end a stack of access control filters.

The layered approach for access control is implemented on the Linux machines in the ATLAS experiment with the Pluggable Authentication Module (PAM) [28] as represented in Figure 23. The RBAC check is integrated in the Linux security mechanism as a PAM module which is used by any PAM aware application to enforce an access policy. The local and remote login (e.g. Secure Shell [29]), and the Sudo [30] tools are a few examples of tools using the PAM.

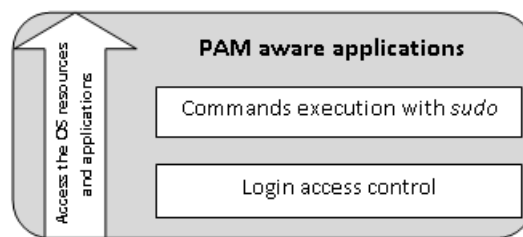


Figure 23 Layered access control on Linux nodes.

The file server machines allow only the users with administrator roles to login and the execution of some commands (e.g. reboot, fdisk) is restricted to the administrator or experts. The online computing nodes can restrict the login access only to the users from the detector group the nodes belong to. These two examples are possible use cases of the layered access control on the Linux nodes.

All these protection levels are detailed later in the dedicated Chapter 5.

3.2.2.2 Control Room Desktop

The Control Room Desktop (CRD) is the Graphical User Interface (GUI) environment available on the desktop machines in the ATLAS Control Room. It is based on the Linux K Desktop Environment and exploits its KIOSK [31] configuration mode. The KIOSK framework provides a set of features to easily and powerfully define and restrict the capabilities of a KDE environment based on the user's credentials. It permits the construction of a controlled environment by customizing and locking almost all the desktop functionalities. The restrictions range from disabling the background wallpaper customization to disabling the user log out button or the possibility to access a command shell. The details on this CRD customization based on access control policies are described later in chapter 5.1.2.

3.2.2.3 TDAQ Access Manager

The management and control aspects of the TDAQ software are covered by the TDAQ Control and Configuration (chapter 10.5 in [10]) software components. The TDAQ Access Manager component is responsible for the access control implementation in the TDAQ software. The component has a client – server architecture where the client sends authorization requests to the server, and the server processes the requests and sends back the responses (authorized or not, the reasons etc) to the client.

The clients are in general TDAQ Controls software components that need to protect their functions based on an access control policy. A few examples are the Process Manager, Run Control,

or IGUI [10]. The authorization decision task is complex and requires communications with the databases holding the RBAC specific data. Another constraint put on the decision taker is to be very fast to not impact the client's performance. The optimal solution is to implement the decision algorithm in a separate software component that is optimized to process and deliver access control decisions as fast as possible. The Access Manager (AM) server accomplishes this task. The clients call functions provided by the AM Client Application Programming Interface (API) to ask for authorization and take the appropriate action according to the response received from the server.

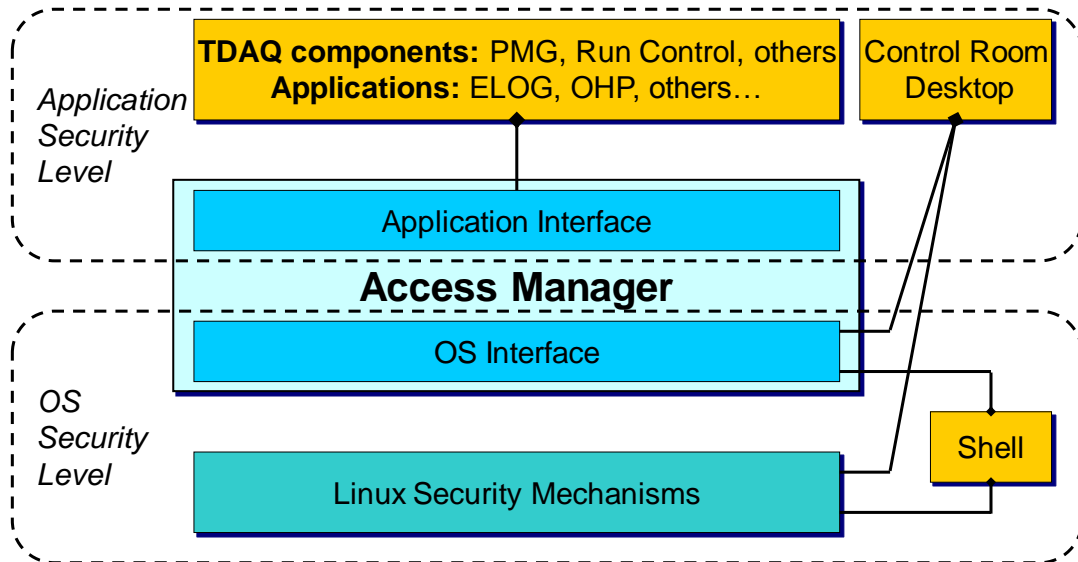


Figure 24 Examples of Access Manager clients.

Other examples of AM Server clients are: the remote access to the ATLAS cluster (the *shell* in Figure 24) and the CRD (more details are later on their chapters). As it can be seen, AM has interfaces to both OS and Application layer, thus assuring a cohesion in the access control enforcements.

The TDAQ Access Manager component is extensively described in its dedicated chapter Chapter 6.

3.2.3 Data flow

3.2.3.1 OASIS XACML perspective

TDAQ AM server component implements the OASIS [32] XACML (eXtensible Access Control Markup Language) standard [13] for its internal policies management and authorization algorithm. The standard defines a declarative access control policy language implemented in XML (Extensible Markup Language) [33] and a processing model describing how to evaluate authorization requests according to the rules defined in policies. An introduction to standard's terminology is given in chapter Chapter 6. What we are interested in at this moment is the data flow defined by the standard and the mapping to our access control solution.

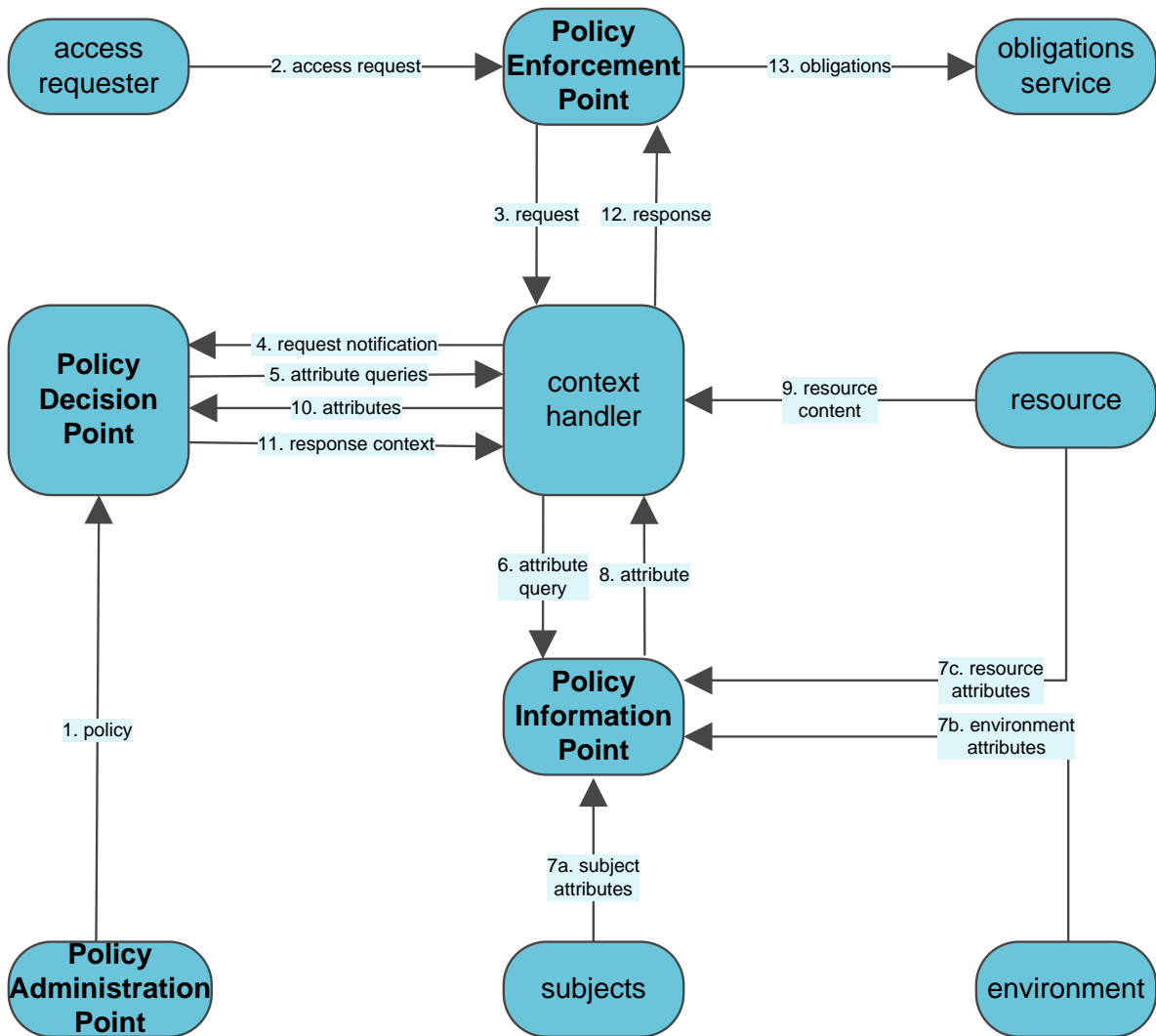


Figure 25 Data flow diagram in the XACML standard.

Figure 25 describes the XACML data flow where the following main actors can be identified:

- Policy Administration Point (PAP): creates and manages the policies definitions.
- Policy Information Point (PIP): acts as a provider of attribute values required in the policy evaluation.
- Policy Decision Point (PDP): evaluates applicable policy and renders an authorization decision.
- Policy Enforcement Point (PEP): performs the access control by making decision requests and enforcing authorization decision.

The access requester may be for example a user who wants to log in through the application gateway inside ATCN, or the TDAQ Process Manager application that is requested by the user to stop a critical application running on a cluster node.

As it can be seen, the main actors described above have clear responsibilities so the mapping to our actors in the access control solution comes naturally and is depicted in Figure 26.

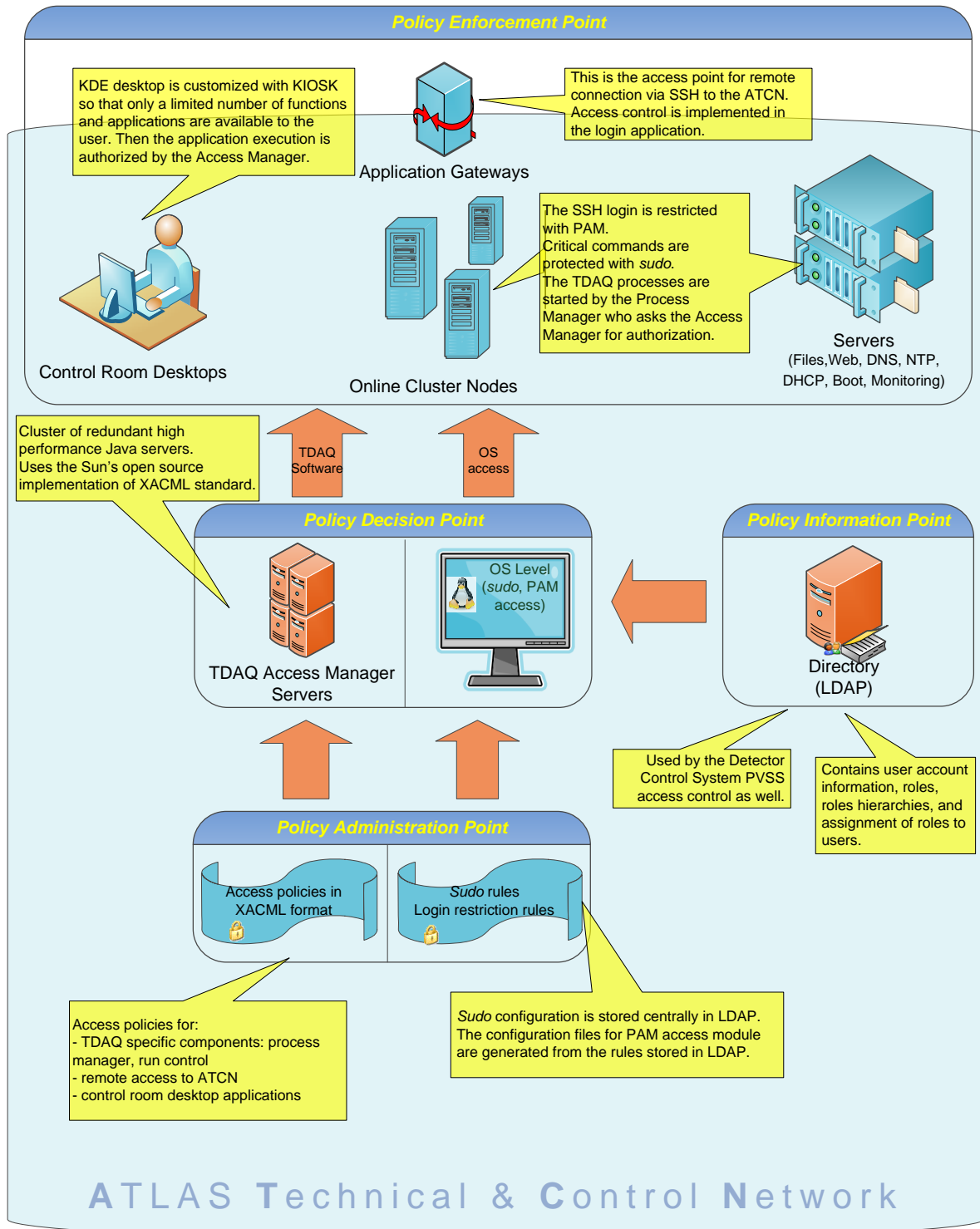


Figure 26 Data flow in the ATLAS access control solution.

Policy Administration Point together with the management of Policy Information Point fall under the RBAC Administration Tool responsibility. The Policy Information Point in turn plays the role of source of information about the users, roles hierarchy and assignment of users to roles.

The Policy Decision Point role is taken by:

- TDAQ AM servers which decides if an account requested by a client is approved or not. The decision is taken based on the permission assignment to role in XACML format and the users/roles information from PIP - Directory server (LDAP).
- OS level specific tools used for restriction on the login and tools execution. Both have configurations (in PAP) based on users and roles information (from PIP).

The enforcements of the decisions provided by PDP are done right at the resource under protection. Hence, the Policy Enforcement Points are:

- The Application Gateways which permit the access to ATCN
- The machine's login (both locally and remote), being them working nodes or servers
- Once logged on a node, the *sudo* tool which executes the application protected by the access control policy
- The TDAQ software components which run on the cluster nodes and execute user's command if approved by PDP (TDAQ AM server in this case)
- The Control Room Desktop where people are provided with the appropriate screen configuration according to the policy checked by PDP
- And the list is open for future integration of TDAQ Software components with TDAQ AM server (e.g. database control)

3.2.3.2 Reference monitor perspective

The data flow in our access control can be viewed also from the perspective of reference monitor concept described in chapter 2.3.4. The mapping to the XACML actors described before is shown in Table 3.

Table 3 Mapping of Reference Monitor concept to XACML model

Reference monitor	XACML model
Subjects	Access requester
Access request	2. access request
Reference monitor	PDP and PEP
Access Control Database	PAP and PIP
Audit file	Obligations service (the logging of enforcement actions like action allowed/denied with reason)
Objects	Resource content

3.3 Security design principles coverage

At the end of this chapter which presented the high level design of the access control solution, we summarize below (Table 4) how the security design principles described in chapter 2.2 are addressed by our solution.

Table 4 Security design principles coverage

Security design principle	How it is addressed
Simplicity	<p>The design is focused on solving the today ATLAS needs in terms of access control: Hierarchical RBAC model is sufficient for the moment, but the extension is always possible to constrained or symmetric RBAC.</p> <p>Our motto in designing the solution is: “Do not re-invent the wheel”</p> <p>XACML standard is used to benefit from the standards organization validation of the concepts and to leave the door open for future integration with systems compliant with XACML.</p> <p>The mechanisms and tools used in the implementation described later are well known in Linux world and their security is continuously validated by the community.</p>
Restriction	The granularity of permissions defined in the policies allows only one operation to be executed on the resource. Hence, this is more a matter of configuration of the solution.
Least privilege	Same as “Restriction”
Fail-Safe defaults	The default role assigned to new users is “Observer” which has a minimum set of permissions. Then, the users can take on more roles after they are trained on new field’s expertise.
Economy of mechanism	Same as “Simplicity”
Complete mediation	Caches are used in the implementation, but a notification mechanism is also set up to propagate the changes to the PDPs.
Open design	We fully describe in this thesis the solution design and implementation details.
Separation of privilege	Even if a user is recognized to have the expertise necessary for a role, the user is allowed to use the role only in a controlled manner (e.g. when the Shift Leader considers necessary the intervention of an expert on a faulty system for a limited time interval).
Least common mechanism	This principle is taken in consideration in the implementation of the access control solution, for example the AM server is designed with overload protection. But the fulfillment of this principle must be extended to the other TDAQ software components which are clients of TDAQ AM server. The later part is not in the scope of this thesis.
Psychological acceptability	The users of the access control solution (physicists, engineers with experience in other CERN experiments) are used to work with group accounts which are mapped now to roles, so the transition to the new access control model is smoother and the resistance to change is minimized.

Chapter 4

RBAC setup and administration

In this chapter we describe how the access control model chosen for ATLAS is set up in the ATLAS Online cluster and the tools developed to manage it.

4.1 Database type

As shown in the previous chapter where the high level solution is outlined (3.2.1.1), a central database is meant to store the RBAC model elements: users, roles, permissions and relationships.

The characteristics of this data to be stored are:

- Data type: the users, roles, permission names are String data types with a short size (the users and roles are at most 50 characters, while the permissions can be a bit longer, but this depends very much on the application where the permissions are enforced)
- Access:
 - Write: the users are defined in the database when the researcher joins CERN, the roles are created or modified with every internal experiment organization change, the permissions are set and associated to roles when the application protected by the permission is installed or updated, and the users are assigned to roles when they expertise level is certified by the organization. As it can be seen, all the write operations on RBAC elements are very rare (order of days) compared with the data write frequencies in computing software (order of seconds, even milliseconds).
 - Read: by contrary, the read operations on this RBAC data are very often and the frequency is given by the needs of applications under protection. The order can be of seconds.
- Availability: RBAC data must be always available because the experiment functioning depends on it. Downtime of database holding RBAC data implies downtime of access control system in general and the critical operations under access control protections can not be performed anymore. It is obvious that the risk of not being able to control the experiment must be avoided.

For all these reasons, we've considered that in fact a LDAP [14] directory service fits [34] the profile of a RBAC model repository. Moreover, the integration of many Linux services with LDAP is out of the box and we can benefit from it in the deployment of the access control solution in the ATLAS Online cluster.

4.2 RBAC setup

As detailed in the previous chapter, the RBAC configuration is centralized in an LDAP directory service in the ATLAS Online cluster. Therefore, all entities specific to Hierarchical RBAC model (users, roles, permissions) and relationships (user assignment to roles, permissions assignment to role, role hierarchies) are defined in an OpenLDAP [35] server hosted on one of the cluster central servers.

The **users** are defined in the LDAP directory as *posixAccount* object class so that the standard PAM [28] authentication on the Linux nodes to be able to recognize them as user accounts on the

node. Figure 27 shows an example of user definition in LDAP as it can be viewed with an LDAP browser (phpLDAPAdmin [36]).

```


  A screenshot of an LDAP browser showing the entry for 'cn=mleahu'. The entry details are as follows:
  dn: cn=mleahu,ou=people,ou=atlas,o=cern,c=ch
  givenName: Marius
  userPassword: {MD5}JCqhqXdpEJBI47TfNZvPyQ==
  sn: Leahu
  gidNumber: 1001
  uid: mleahu
  uidNumber: 1001
  objectClass: inetOrgPerson, posixAccount, top
  cn: mleahu
  homeDirectory: /home/mleahu
  
```

Figure 27 User definition in LDAP













	dn	objectClass	cn	nisNetgroupTriple	memberNisNetgroup
	cn=RA-DCS:expert,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RA-DCS:expert	(,bob,)	
	cn=RA-DCS:shifter,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RA-DCS:shifter		RA-ShiftLeader RA-DCS:expert
	cn=RA-Observer,ou=netgroup,ou=atlas,o=ce...	top nisNetgroup amRole	RA-Observer		RA-TDAQ:shifter RA-DCS:shifter
	cn=RA-ShiftLeader,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RA-ShiftLeader	(,alice,)	
	cn=RA-TDAQ:expert,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RA-TDAQ:expert		
	cn=RA-TDAQ:shifter,ou=netgroup,ou=atlas,...	top nisNetgroup amRole	RA-TDAQ:shifter	(,mleahu,)	RA-TDAQ:expert RA-ShiftLeader
	cn=RE-DCS:expert,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RE-DCS:expert		
	cn=RE-DCS:shifter,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RE-DCS:shifter		RE-ShiftLeader RE-DCS:expert
	cn=RE-Observer,ou=netgroup,ou=atlas,o=ce...	top nisNetgroup amRole	RE-Observer		RE-TDAQ:shifter RE-DCS:shifter
	cn=RE-ShiftLeader,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RE-ShiftLeader	(,alice,)	
	dn	objectClass	cn	nisNetgroupTriple	memberNisNetgroup
	cn=RE-TDAQ:expert,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RE-TDAQ:expert		
	cn=RE-TDAQ:shifter,ou=netgroup,ou=atlas,...	top nisNetgroup amRole	RE-TDAQ:shifter	(,mleahu,)	RE-TDAQ:expert RE-ShiftLeader

Figure 28 Roles and role hierarchy definitions in LDAP

The **roles** are mapped in LDAP as *nisNetgroup* object class and *amRole* object class. The use of NIS netgroups [37] brings the following advantages:

- Out of the box integration with many Linux tools (e.g. *Sudo* [30]) and mechanisms (e.g. various PAM modules are able to work with netgroups), hence the roles are seen as natural components in Linux environment
- Aggregation of more netgroups in one netgroup. This is very helpful to set up the role hierarchies and permission inheritance over the hierarchy.

We defined a dedicated LDAP schema [38] to be used by OpenLDAP instances to extend netgroups definitions with details necessary for the RBAC model. The appendix 8.1 details the schema specification. The *amRole* object class defined in this schema brings the following enhancements:

- Label the NIS netgroups defined in LDAP with the “amRole” qualifier for easier differentiation in LDAP queries
- Attach more properties to netgroups to be used in more advanced RBAC models definition, such as:
 - if the role is assignable to users or not (in that case, the role is just internally in the hierarchy just to allow permissions sharing)
 - future constraints like Static Separation of Duties or Dynamic Separation of Duties.

Figure 28 shows the role definition in LDAP as netgroups for the example of role hierarchy from Figure 21. The columns from the figure are described in Table 5.

Table 5 Roles definition in LDAP

Object attribute	Description
dn	the “path” to the object in the LDAP tree
objectClass	specifies a set of attributes used to describe an object; in our case, the object classes described above are mandatory
cn	the role name prefixed with RA or RE. There are two netgroups defined in LDAP for each role: <ul style="list-style-type: none"> - RA-<rolename> shows that the <rolename> is assigned to an user - RE-<rolename> shows that the <rolename> is enabled for an user during its session (see 3.2.1.1.2 for details on user sessions) The roles not assignable to user (see 3.2.1.1.1 for details) are prefixed with RN.
nisNetgroupTriple	used for user assignment relationship
memberNisNetgroup	used for role hierarchy relationship

The **permissions** and **permission assignment** relations are specific to the applications enforcing them, so we will describe them in the following chapters dedicated to each enforcement type.

The **user assignment to roles** is accomplished by setting role’s netgroup property *nisNetgroupTriple* to the user name in the format (*,<username>*). The roles assigned to a user have 2 states:

- assigned: this is the initial state when the user is certified for the role. For example, a user who graduated training on DCS technology is recognized as expert in DCS department, hence he gets the DCS expert role assigned.
- enabled: a role already assigned to a user is enabled so that the user is able to perform the tasks allowed by this role. This corresponds to a user session as described in 3.2.1.1.2. The user with DCS expert role can work on DCS hardware only when his role is enabled. In this way, the exclusive access to resources can be regulated by the group leaders.

The example in Figure 28 has three users with the following roles assignment:

- roles assigned: mleahu – TDAQ:shifter, alice – ShiftLeader, bob – DCS:expert
- roles enabled: mleahu – TDAQ:shifter, alice – ShiftLeader, bob – no roles enabled.

The **role hierarchy** is mapped to the netgroup aggregation relationship. This is configured in LDAP thanks to the *memberNisNetgroup* attribute of a role: the value of this attribute represents a senior role for the current role. For example in Figure 21, the role DCS shifter has senior roles ShiftLeader and DCS expert, both represented as values of its memberNisNetgroup attribute; on the other hand, DCS shifter role is senior for Observer, hence Observer's memberNisNetgroup value contains it. The role hierarchy managed through netgroups permits **permission inheritance** from junior to senior roles. This means that permission assigned to role Observer is allowed to users with the role Observer and all Observer's seniors (direct or indirect).

4.2.1 OpenLDAP service configuration

As mentioned in the previous chapter, the OpenLDAP is used as LDAP service in the ATLAS online cluster. The central servers that host this service run the CERN's Linux version, namely Scientific Linux CERN 5 (SLC 5) [23] based on the Red Hat Enterprise 5 Linux version. In order to configure the RBAC in LDAP directory as described above, the following configuration checks or adjustments are necessary after the installation of OpenLDAP package on SLC 5 server:

- `nis.schema` must be included in the server configuration file (`slapd.conf`). This schema is necessary to support the NIS netgroups information in the LDAP structure [39].
- `sudo.schema` [40] (shipped also in the `sudo` package in `/usr/share/doc/sudo-1.7.2p1/schema.OpenLDAP`) must be included in the server configuration to allow the definitions of sudo permissions (more details in the chapter 5.2.2).
- `amRBAC.schema` must be also included in the server configuration. Its content is included in the Appendix section 8.1.

4.3 RBAC administration tool

The RBAC Administration tool described in 3.2.1.1 addresses a large spectrum of functionalities meant to cover the RBAC model administration in the ATLAS Online cluster. We identified the requirements to be fulfilled by the RBAC Administration tool and these are presented in the following chapter. Our worked focused first on the top priority needs from the RBAC Administration tool and, at this stage, they are fulfilled by a set of shell scripts and web interface for LDAP management. The requirements coverage matrix is filled in the last chapter of this section. The development of a unique tool to gather under its umbrella all RBAC administration functions is an action point on the "future work" list.

4.3.1 Requirements

The requirements for the RBAC Administration tool are gathered in a structured document "ATLAS RBAC Administration Tool Requirements" which is included in this thesis in the Appendices

chapter, 8.2. The document contains the requirements organized by use cases split over 3 main categories detailed in the following sub chapters. The actors in the use cases are of two types:

- administrators: define the roles, the access policies and assign/revoke the roles to/from users.
- shifters: enable and disable the roles already assigned to users by the administrators.

4.3.1.1 Administrative functions

The administrative functions refer to creation and maintenance of elements sets and relations for building the RBAC model. Figure 29, Figure 30, Figure 31 and Figure 32 summarize the use cases which fall in the administrative category:

- Roles administration
- Permissions administration
- Relations administration:
 - Role hierarchy (RH)
 - User assignment (UA)
 - Permission Assignment (PA)
 - Static Separation of Duties(SSD)/Dynamic Separation of Duties(DSD)

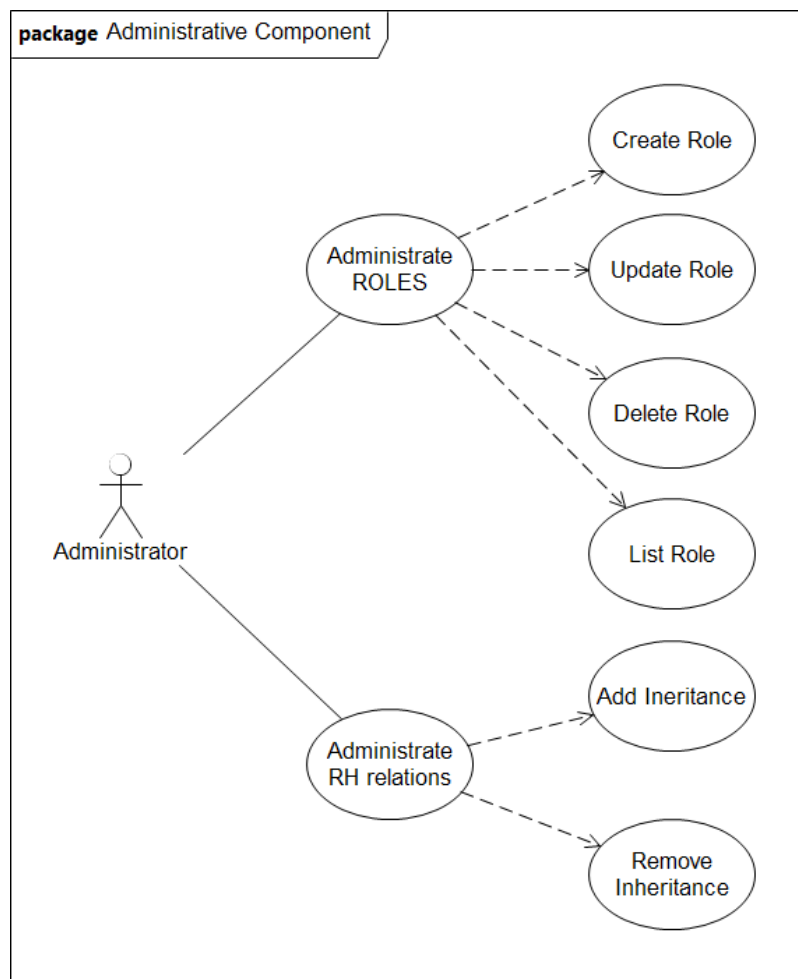


Figure 29 Administrative Component use cases - Roles Management

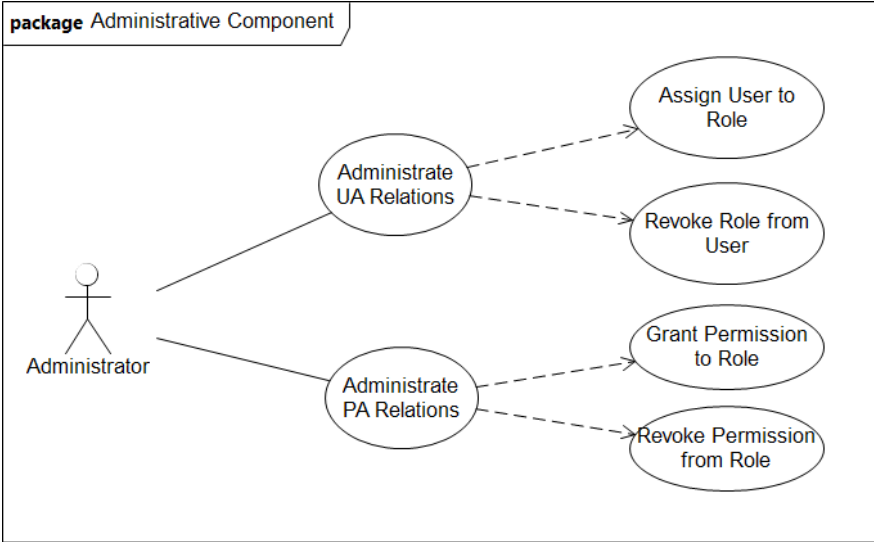


Figure 30 Administrative Component use cases - Users, Permissions Assignments to Roles

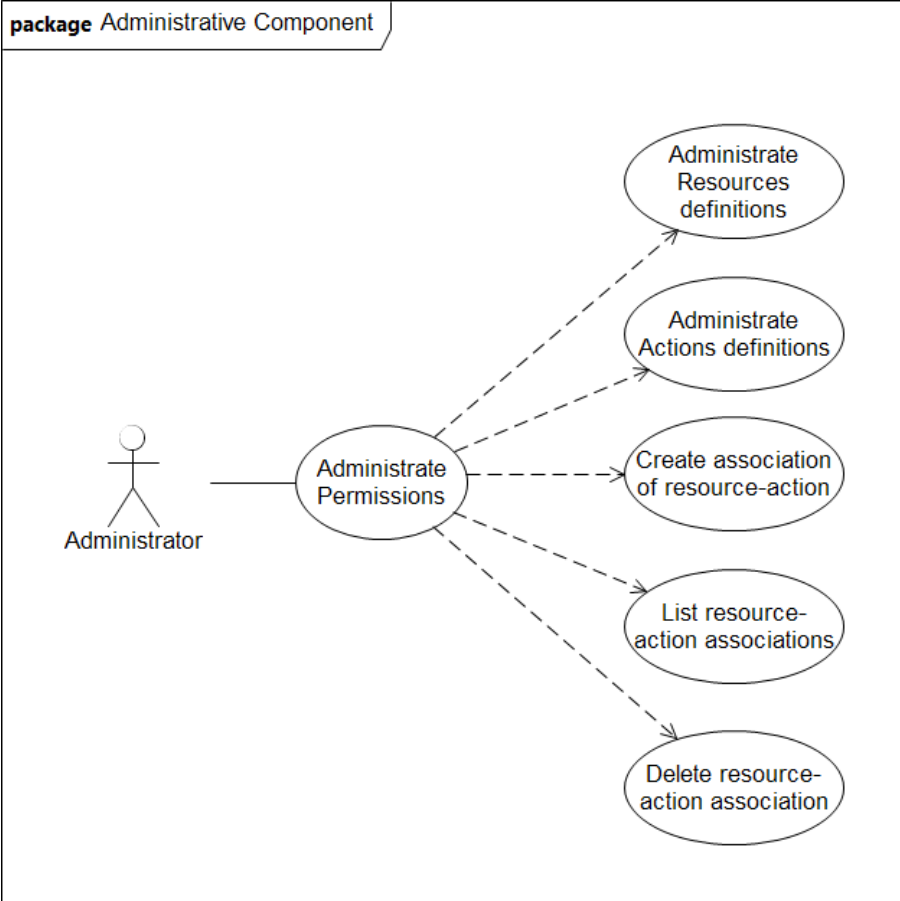


Figure 31 Administrative Component use cases - Permissions

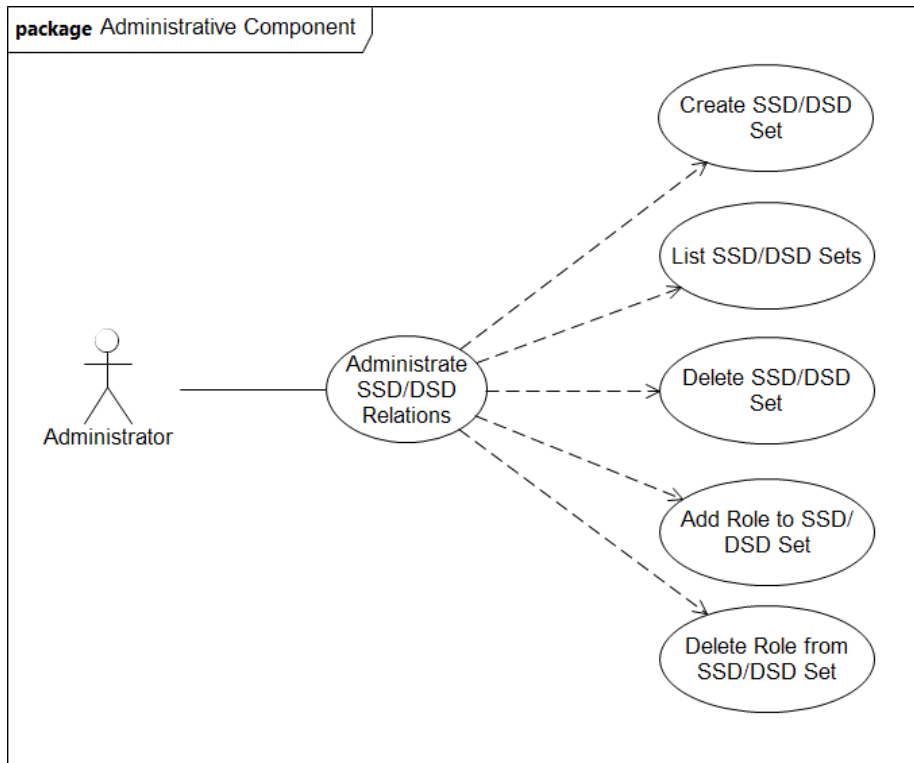


Figure 32 Administrative Component use cases - Separation of Duties

Note that users management is not in the scope of RBAC Administration tool and this point is tackled in 4.3.2.

4.3.1.2 User session management

The user session management addresses the need of having the users acting with their role in a controlled manner. Certain roles (e.g. expertise on sensitive hardware) may need a careful management and the actions allowed by them may need to be approved also by the Shifter user (the users who monitor the experiment functioning). The user session represents the time interval when the user's role is enabled, hence he can perform action allowed by the role.

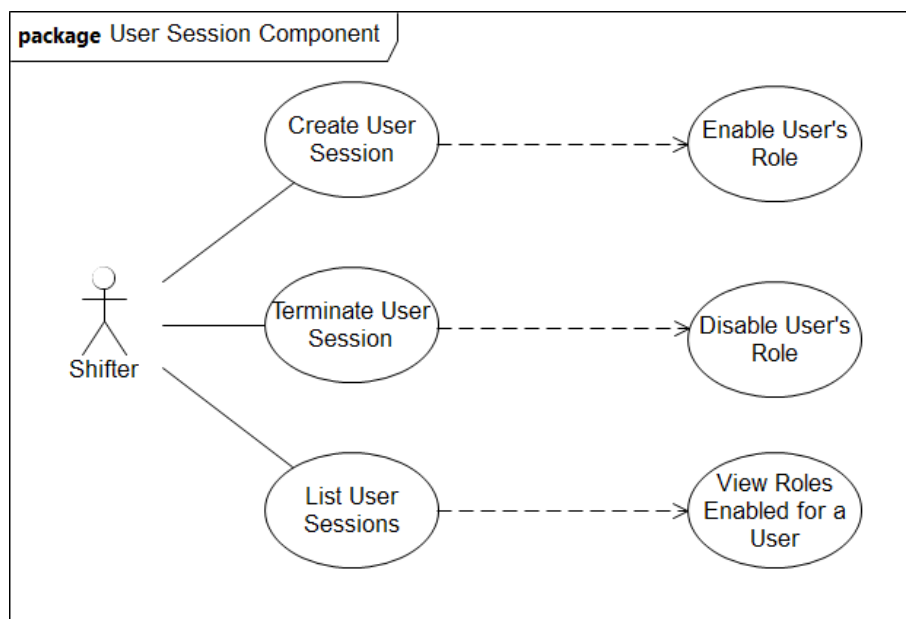


Figure 33 Use Case diagram for Users Sessions management

4.3.1.3 Review functions

The results of the actions created by the administrative functions are reviewed thanks to the “Review functions”. The use cases addressed by this category of functions are represented in Figure 34.

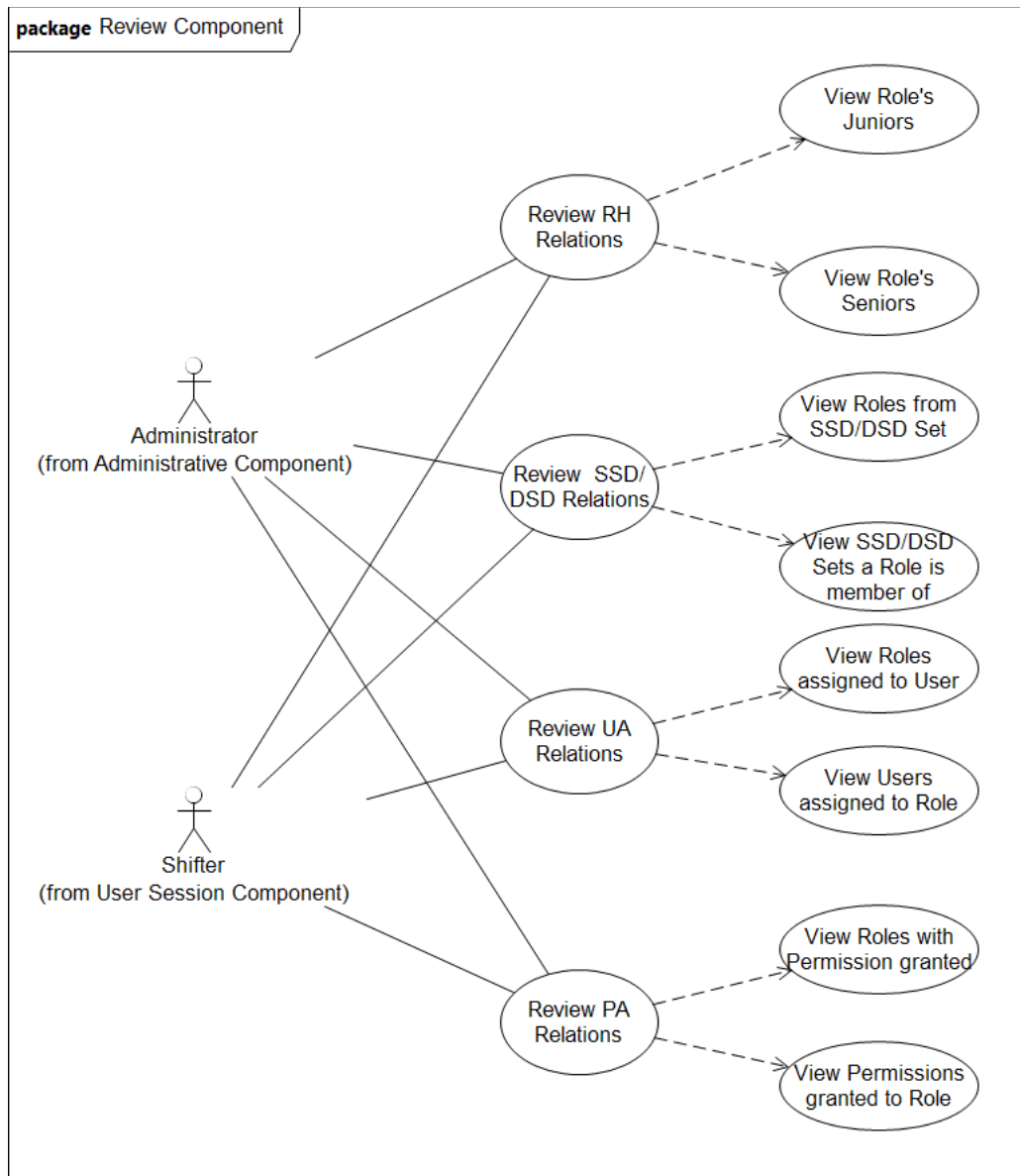


Figure 34 Use Case Diagram for Review Component

4.3.2 User accounts

The CERN users and implicitly the ATLAS experiment users are administrated by the CERN Human Resources (HR) department in a centralized HR database. The department is responsible for keeping the users information up to date, so the HR database is the reference database for the CERN user computing accounts database. The CERN IT department administrates the CERN user computing accounts and keeps it synchronized with the HR database. At the same time, one of the ATLAS experiment requirements is to be able to run and take data for up to 24 hours while disconnected from the CERN IT department. Consequently, the direct use of the CERN user computing accounts database by the ATLAS computing system would make the experiment impossible to run. The solution [22] is to mirror the ATLAS user accounts from the CERN user computing accounts database in a database located in the experimental area, more specifically, the LDAP directory service running

on the central file servers. The management of ATLAS user accounts and their synchronization with CERN IT is performed by TDAQ SysAdmin specific tools not addressed exhaustively in our thesis.

4.3.3 Tools

During the activities performed to set up the access control solution in the ATLAS online computing cluster, we developed a set of tools to fulfill the majority of requirements mentioned in the previous paragraph. The access to LDAP server is facilitated by the LDAP command line tools available on SLC in the `openldap-clients` package:

```
/usr/bin/ldapadd
/usr/bin/ldapdelete
/usr/bin/ldapmodify
/usr/bin/ldapsearch
```

As Graphical User Interface (GUI) for the LDAP server administration, we used the *phpLDAPAdmin* web application also available in the SLC 5 distribution (package `phpldapadmin`).

The *write* operations on the RBAC model in the LDAP server are allowed to be performed by the `AccessManagerAdmin` user account. The *read* operations are allowed to everybody.

The following sub chapters detail the tools functionalities.

4.3.3.1 Roles management

We developed a bash [41] shell script to manage the **roles** and **roles hierarchies** in LDAP: `amRolesManager` (listing available in appendix 0). The summary of its functionalities are shown in its help screen:

```
# ./amRolesManager -h
amRolesManager, version $Revision: 52065 $
Access Manager utility to administrate roles and roles hierarchies.

Usage: amRolesManager [-c|-C|-d|-u "(attribute=value)"|[-s|-S senior]|[-j|-J
junior]|-L] <-r ROLE> [-m userBindDN] [-M] [-p password] [-l ldapserver] [-b
basedn] [-v] [-f] [-h]
No arguments will make the script display all the roles found in LDAP.
  -c create the role ROLE not assignable to users
  -C create the role ROLE assignable to users
  -d delete the role ROLE
  -u update the ROLE attribute with a new value. The format must be
' (attribute1=value1) (attribute2=value2) '
  -s add a senior role for the ROLE
  -S remove a senior role from the ROLE
  -j add a junior role for the ROLE
  -J remove a junior role from the ROLE
  -L dump the LDIF for role
  -r the ROLE name; if no other script argument provided, the ROLE details are
displayed.
  -m authenticate as <userBindDN> to the LDAP server; default is AccessManagerAdmin
  -M authenticate as [cn=Manager] to the LDAP server
  -p password to bind to the LDAP server
  -l use this ldapserver;          default is [localhost]
  -b use this basedn;              default is [ou=atlas,o=cern,c=ch]
  -f force mode; no confirmation asked for change operations
  -v verbose mode
  -h this info

Author: mleahu@CERN
```

In order to prepare the roles hierarchy from example in Figure 21, the following commands are executed with the `amRolesManager` script:

- create all roles as assignable to users:

```
#!/amRolesManager -C -r "Observer TDAQ:shifter DCS:shifter TDAQ:expert DCS:expert ShiftLeader"
>>> Create ASSIGNABLE role [Observer]...>>> DONE!
>>> Create ASSIGNABLE role [TDAQ:shifter]...>>> DONE!
>>> Create ASSIGNABLE role [DCS:shifter]...>>> DONE!
>>> Create ASSIGNABLE role [TDAQ:expert]...>>> DONE!
>>> Create ASSIGNABLE role [DCS:expert]...>>> DONE!
>>> Create ASSIGNABLE role [ShiftLeader]...>>> DONE!
```

- Set the relationships between Observer (junior) and TDAQ:shifter, DCS:shifter (seniors):

```
# ./amRolesManager -j Observer -r "TDAQ:shifter DCS:shifter"
>>> add juniors for role [TDAQ:shifter]...
...junior [Observer]:OK
modifying entry "cn=RA-Observer,ou=netgroup,ou=atlas,o=cern,c=ch"

modifying entry "cn=RE-Observer,ou=netgroup,ou=atlas,o=cern,c=ch"

>>> DONE!
>>> add juniors for role [DCS:shifter]...
...junior [Observer]:OK
modifying entry "cn=RA-Observer,ou=netgroup,ou=atlas,o=cern,c=ch"

modifying entry "cn=RE-Observer,ou=netgroup,ou=atlas,o=cern,c=ch"

>>> DONE!
```

- Set the relationships between ShiftLeader (senior) and TDAQ:shifter, DCS:shifter (juniors):

```
# ./amRolesManager -s ShiftLeader -r "TDAQ:shifter DCS:shifter"
>>> add seniors for role [TDAQ:shifter]...
...senior [ShiftLeader]:OK
modifying entry "cn=RA-TDAQ:shifter,ou=netgroup,ou=atlas,o=cern,c=ch"

modifying entry "cn=RE-TDAQ:shifter,ou=netgroup,ou=atlas,o=cern,c=ch"

>>> DONE!
>>> add seniors for role [DCS:shifter]...
...senior [ShiftLeader]:OK
modifying entry "cn=RA-DCS:shifter,ou=netgroup,ou=atlas,o=cern,c=ch"

modifying entry "cn=RE-DCS:shifter,ou=netgroup,ou=atlas,o=cern,c=ch"

>>> DONE!
```

- Set the relationships between TDAQ:expert (senior) and TDAQ:shifter (junior):

```
# ./amRolesManager -s TDAQ:expert -r "TDAQ:shifter"
>>> add seniors for role [TDAQ:shifter]...
...senior [TDAQ:expert]:OK
modifying entry "cn=RA-TDAQ:shifter,ou=netgroup,ou=atlas,o=cern,c=ch"

modifying entry "cn=RE-TDAQ:shifter,ou=netgroup,ou=atlas,o=cern,c=ch"

>>> DONE!
```

- Set the relationships between DCS:expert (senior) and DCS:shifter (junior):

```
# ./amRolesManager -s DCS:expert -r "DCS:shifter"
>>> add seniors for role [DCS:shifter]...
...senior [DCS:expert]:OK
modifying entry "cn=RA-DCS:shifter,ou=netgroup,ou=atlas,o=cern,c=ch"

modifying entry "cn=RE-DCS:shifter,ou=netgroup,ou=atlas,o=cern,c=ch"

>>> DONE!
```

If desired, an additional role not assignable can be defined to aggregate permissions on the usage of DAQ expert tools. This role is inherited by TDAQ:expert and can be inherited also by other DAQ expert roles (e.g. LAR:DAQ:expert etc). The commands are:

```
# ./amRolesManager -c -r "DAQ:expert:tools"
>>> Create NOT ASSIGNABLE role [DAQ:expert:tools]...>>> DONE!
# ./amRolesManager -s TDAQ:expert -r "DAQ:expert:tools"
>>> add seniors for role [DAQ:expert:tools]...
...senior [TDAQ:expert]:OK
modifying entry "cn=RA-DAQ:expert:tools,ou=netgroup,ou=atlas,o=cern,c=ch"

modifying entry "cn=RE-DAQ:expert:tools,ou=netgroup,ou=atlas,o=cern,c=ch"

>>> DONE!
```

A verbose display of the roles defined so far can be obtained by running:

```
# ./amRolesManager -v
DISPLAY THE ROLE INFORMATION
### 7 ROLES IN LDAP [ DAQ:expert:tools DCS:expert DCS:shifter Observer ShiftLeader
TDAQ:expert TDAQ:shifter ]
#----- DETAILS -----#
##### ROLE [DAQ:expert:tools] #####
amRoleAssignable: false
JUNIORS(0)
SENIORS(1) TDAQ:expert

##### ROLE [DCS:expert] #####
amRoleAssignable: true
JUNIORS(1) DCS:shifter
SENIORS(0)

##### ROLE [DCS:shifter] #####
amRoleAssignable: true
JUNIORS(1) Observer
SENIORS(2) DCS:expert ShiftLeader

##### ROLE [Observer] #####
amRoleAssignable: true
JUNIORS(0)
SENIORS(2) DCS:shifter TDAQ:shifter

##### ROLE [ShiftLeader] #####
amRoleAssignable: true
JUNIORS(2) DCS:shifter TDAQ:shifter
SENIORS(0)

##### ROLE [TDAQ:expert] #####
amRoleAssignable: true
JUNIORS(2) DAQ:expert:tools TDAQ:shifter
SENIORS(0)

##### ROLE [TDAQ:shifter] #####
amRoleAssignable: true
JUNIORS(1) Observer
SENIORS(2) ShiftLeader TDAQ:expert
```

A subset of roles can also be looked up with a filter. For example, the list of roles with names starting with *TDAQ* is retrieved by running:

```
# ./amRolesManager -r "TDAQ*"
### Found more roles with name [TDAQ*]: 2 roles #####
##### ROLE [TDAQ:expert] #####
amRoleAssignable: true
JUNIORS(2) DAQ:expert:tools TDAQ:shifter
SENIORS(0)
```

```
##### ROLE [TDAQ:shifter] #####
amRoleAssignable: true
JUNIORS(1) Observer
SENIORS(2) ShiftLeader TDAQ:expert
```

The visual display of roles and roles hierarchies is possible thanks to a PHP [42] script that we developed for integration in administration pages of ATLAS Online cluster. The listing of the script is included in the appendix 8.4. The script algorithm consists into following steps:

- Retrieve roles from LDAP according to an input filter
 - For example, the roles which don't belong to DAQ can be retrieved by inserting the following script call into an HTML page:

```

```

- Prepare the list of nodes to be fed to the script which generates the visual tree: `GDRenderer.php` [43]. One more filter can be applied here too: the top role names to match a certain regular expression.
 - For example, from the list of roles retrieved from LDAP, display only the trees with the root containing DAQ. The script call in this case is:

```

```

Only the tree hierarchies are displayed by this script, hence the roles hierarchy from commands executed in the previous chapter is shown in three distinct trees: one tree corresponding to each top role.

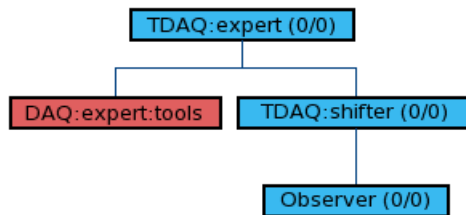


Figure 35 The role tree for TDAQ:expert role.

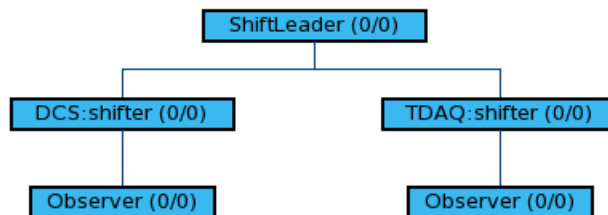


Figure 36 The role tree for ShiftLeader role.

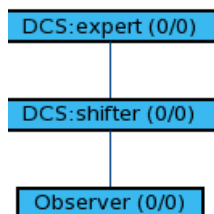


Figure 37 The role tree for DCS:expert role.

Assignable roles are depicted in blue with the number of users having the role enabled/assigned. The roles not assignable to users are colored in red.

4.3.3.2 User's roles management

The management of user assignment to roles relationships is handled by the `amUserRoles` shell script (its listing is included in appendix 8.5). Its functions are summarized in its help screen:

```
./amUserRoles -h
amUserRoles, version $Revision: 52762 $
Access Manager script to manage the users-roles RBAC relationship.

Usage: amUserRoles [-h] [-G role_category] [-L] [-a|-A|-r|-e|-d|-c|-C|-Q roles] [-s|-S|-D] [-f] [-u usernames] [-F input_file] [-m userBindDN] [-M] [-p password] [-l ldapserver] [-b basedn] [-v]
-h this info

-G <role_category> to be used as prefix for the <roles>;

Roles operations (only ONE operation from the list below can be called):
-L list the roles defined in LDAP in the role_category (if specified)
-a assign the <roles> to the <username>
-A assign and enable the <roles> to the <username>
-r revoke the <roles> for the <username>
-e enable the <roles> for the <username>
-d disable the <roles> for the <username>
-c display the users with the <roles> assigned
-C display the users with the <roles> assigned and enabled
-Q display the users with the <roles> assigned and disabled
-s show the roles assigned to the user
-S show the roles assigned and enabled to the user
-D show the roles assigned and disabled for the user

-f display the roles in a format to be stored in files
-u user names; default user is [root]
-F input text file with usernames one per line at the beginning; comments start with #
-m authenticate as <userBindDN> to the LDAP server; default is AccessManagerAdmin
-M authenticate as [cn=Manager] to the LDAP server
-p password to bind to the LDAP server
-l use this ldapserver;          default is [localhost]
-b use this basedn;              default is [ou=atlas,o=cern,c=ch]
-v verbose mode

Author: mleahu@CERN
```

The role assignment to users for the example in Figure 28 can be achieved by running the following commands:

```
# ./amUserRoles -A "TDAQ:shifter" -u mleahu
>>> Assign and enable roles to user [mleahu]...
>>> ...role [TDAQ:shifter]...assign >>> OK! ...enable >>> OK!

# ./amUserRoles -a "ShiftLeader" -u alice
>>> Assign roles to user [alice]...
>>> ...role [ShiftLeader]...assign >>> OK!
# ./amUserRoles -e "ShiftLeader" -u alice
>>> Enable roles for user [alice]...
>>> ...role [ShiftLeader]...enable >>> OK!

# ./amUserRoles -a "DCS:expert" -u bob
>>> Assign roles to user [bob]...
>>> ...role [DCS:expert]...assign >>> OK!
```

The script allows also the following lookups:

- roles assigned (and enabled) directly to a user:

```
# ./amUserRoles -s -u mleahu -v
Check user names against LDAP...
```

```
### DISPLAY USER[mleahu]'S ROLES IN STATE [ASSIGNED] ###
TDAQ:shifter

# ./amUserRoles -S -u bob -v
Check user names against LDAP...
### DISPLAY USER[bob]'S ROLES IN STATE [ENABLED] ###
```

- the user with a given role assigned (and enabled) directly to them:

```
./amUserRoles -C ShiftLeader -v
DISPLAY THE CONTENT OF ROLES [ShiftLeader] IN STATE [ENABLED]
alice

# ./amUserRoles -c DCS:expert -v
DISPLAY THE CONTENT OF ROLES [DCS:expert] IN STATE [ASSIGNED]
bob

# ./amUserRoles -C "ShiftLeader TDAQ:expert TDAQ:shifter" -v
DISPLAY THE CONTENT OF ROLES [ShiftLeader TDAQ:expert TDAQ:shifter] IN STATE
[ENABLED]
### ROLE [ShiftLeader]
alice
### ROLE [TDAQ:expert]
### ROLE [TDAQ:shifter]
mleahu
```

In order to get the list of users which are belong directly or indirectly (by inheritance) to a certain role, the following system commands which interrogate directly the role's netgroup content can be used:

```
# getent netgroup RA-Observer
RA-Observer      ( , bob, ) ( , alice, ) ( , mleahu, )

# getent netgroup RE-Observer
RE-Observer      ( , alice, ) ( , mleahu, )

# getent netgroup RA-TDAQ:shifter
RA-TDAQ:shifter  ( , mleahu, ) ( , alice, )

# getent netgroup RA-DCS:shifter
RA-DCS:shifter   ( , bob, ) ( , alice, )

# getent netgroup RE-DCS:shifter
RE-DCS:shifter   ( , alice, )
```

The roles can be easily disabled or revoked by calling the `amUserRoles` script:

```
# ./amUserRoles -d "TDAQ:shifter Observer" -u mleahu
>>> Disable roles for user [mleahu]...
>>> ...role [TDAQ:shifter]...disable >>> OK!
>>> ...role [Observer]...SKIPPED! Role not assigned!

# ./amUserRoles -r "ShiftLeader Observer" -u alice
>>> Revoke roles from user [alice]...
>>> ...role [ShiftLeader]...disable >>> OK! ...revoke >>> OK!
>>> ...role [Observer]...SKIPPED! Role not assigned!
```

4.3.3.3 Permissions management

The permissions are very specific to the application types being protected and how they are processed in the access control solution: either by the operating system mechanism or by the TDAQ Access Manager server.

In the case of permissions enforced with the help of OS mechanisms (e.g. `sudo`, `PAM`), the permissions configuration is performed in LDAP through the web interface. The LDAP web

administration interface is offered by the phpLDAPAdmin [36] package available in SCL 5 distribution. More details on its usage will be provided later in the chapter Chapter 5.

The TDAQ Access Manager component operates with the access control policies in XACML [13] format stored in files. However, the permissions for AM clients are defined in a simplified format in files which are then transformed in XACML format by a dedicated tool. The format of this file and the tool to be used to generate XACML policies are addressed in Chapter 6.

4.3.4 Requirements coverage matrix

After we presented the tools put in place to manage the RBAC model, we summarize the coverage of the RBAC administration tool requirements in the tables below.

Requirement ID	Requirement Title	Priority	Fulfilled	Observations
URA01	Role Creation	High	Yes	<code>amRolesManager -C</code>
URA02	Role Update	High	Yes	<code>amRolesManager -u</code>
URA03	Role Deletion	High	Yes	<code>amRolesManager -d</code>
URA04	Role Listing	High	Yes	<code>amRolesManager [-r]</code>
URA05	Role Inheritance Creation	High	Yes	<code>amRolesManager [-s -j]</code>
URA06	Role Inheritance Deletion	High	Yes	<code>amRolesManager [-S -J]</code>
URA07	Senior Role Creation	High	Yes	<code>amRolesManager -C</code> && <code>amRolesManager -s</code>
URA08	Junior Role Creation	High	Yes	<code>amRolesManager -C</code> && <code>amRolesManager -j</code>
URA09	SSD Set Creation	Low	No	Not in the current scope
URA10	SSD Set Deletion	Low	No	Not in the current scope
URA11	SSD Set Listing	Low	No	Not in the current scope
URA12	SSD Role Member Addition	Low	No	Not in the current scope
URA13	SSD Role Member Deletion	Low	No	Not in the current scope
URA14	DSD Set Creation	Low	No	Not in the current scope
URA15	DSD Set Deletion	Low	No	Not in the current scope
URA16	DSD Set Listing	Low	No	Not in the current scope
URA17	DSD Role Member Addition	Low	No	Not in the current scope
URA18	DSD Role Member Deletion	Low	No	Not in the current scope
URA19	Resource Instance Creation	High	Yes	See permission definition
URA20	Resource Instance Update	High	Yes	See permission definition
URA21	Resource Instance Deletion	High	Yes	See permission definition
URA22	Resource Instance Listing	High	Yes	See permission definition
URA23	Action Instance Creation	High	Yes	See permission definition

URA24	Action Instance Deletion	High	Yes	See permission definition
URA25	Action Instance Listing	High	Yes	See permission definition
URA26	Permission Creation	High	Yes	See permission definition
URA27	Permission Deletion	High	Yes	See permission definition
URA28	Permission Listing	High	Yes	See permission definition
URA29	Role Assignment to Users	High	Yes	<code>amUserRoles -a</code>
URA30	Role Revocation from Users	High	Yes	<code>amUserRoles -r</code>
URA31	Grant Permission to Role	High	Yes	See permission definition
URA32	Revoke Permission from Role	High	Yes	See permission definition
URA33	Relationships Integrity	Normal	No	
URA34	Selection Filters	Normal	No	
URUS01	User's Role Enabling	Normal	Yes	<code>amUserRoles -e</code>
URUS02	User's Role Disabling	Normal	Yes	<code>amUserRoles -d</code>
URUS03	View Roles Enabled for User	Normal	Yes	<code>amUserRoles -S</code>
URUS04	View Roles Disabled for User	Normal	Yes	<code>amUserRoles -D</code>
URUS05	View Users with a Role Enabled	Low	Yes	<code>amUserRoles -C</code>
URUS06	View Users with a Role Disabled	Low	Yes	<code>amUserRoles -Q</code>
URR01	View Role's Juniors	Normal	Yes	<code>amRolesManager -r</code>
URR02	Graphical Display of Role's Juniors	Low	No	Not in the current scope
URR03	View Role's Seniors	Normal	Yes	<code>amRolesManager -r</code>
URR04	Graphical Display of Role's Seniors	Low	No	Not in the current scope
URR05	View Roles from a SSD Set	Low	No	Not in the current scope
URR06	View Roles from a DSD Set	Low	No	Not in the current scope
URR07	View the SSD Sets a Role is Member of	Low	No	Not in the current scope
URR08	View the DSD Sets a Role is Member of	Low	No	Not in the current scope
URR09	View Roles Assigned to User	Normal	Yes	<code>amUserRoles -s</code>
URR10	View Users Assigned to Role	Normal	Yes	<code>amUserRoles -c</code>
URR11	View Permissions Granted to Role	Normal	Yes	See permission definition
URR12	View Permissions Granted to User	Low	No	
URR13	View Roles with Permission	Normal	Yes	See permission definition

	Granted			
URR14	View Users with Permission Granted	Low	No	
URR15	Search Permissions	Normal	Partially	See permission definition

Requirement ID	Requirement Title	Priority	Fulfilled	Observations
URPER01		Normal	Yes	
URPOR01	Operating System	Normal	Yes	
URI01	Graphical User Interface	Low	No	Not in the current scope
URI02	Command Line Interface	Normal	Partially	Permission management is not fully covered.
URO01	Remote Operation	Normal	Yes	
URS01	Self Protection	Normal	Yes	
URS01	Operation Traceability	Normal	Partially	Not all tools log messages.
URRES01	Hardware Requirements	Normal	Yes	Runs on standard hardware in ATLAS Online cluster
URRES01	Software Requirements	Normal	Yes	
URD01	User Manual	Normal	Partially	Documentation must be structured in a dedicated manual for users. Currently available in the SysAdmin FAQ (Frequently Asked Questions) web page.

Chapter 5

Access control at the system administration level

The ATLAS Online cluster has the requirement to be able to run the experiment and take data for up to 24 hours while having lost the connection to the IT department and database centre (responsible for long term storage of the data). Given these requirements, the cluster is designed to be as autonomous as possible with its own cluster services (DNS, NTP, LDAP, File servers etc) and isolated from direct internet access. The users can access the cluster by two means depending on their geographical location:

- Users not in the ATLAS site premises can access remotely the ATLAS Online cluster by hopping through an Application Gateway. These users are detectors and sub detectors experts located sometimes at their universities or at home. They need access to the computers connected to their detector hardware to debug problems or adjust configurations. In order to reach the cluster nodes, the users must go through the following steps:
 - the users must go first to the CERN Public Linux service (LXPLUS [44])
 - from there, they can open a secure shell connection (*ssh* [29]) to the ATLAS Application gateway (atlasgw.cern.ch) where they are invited to type in the destination node within the ATLAS Online cluster
- Users present in the ATLAS Control Room (ACR) [45] (Figure 38) can access the desktops installed in the control room from where they can monitor and control detector hardware or the running of acquisition software. Each detector has its own desk in the control room and the experts and shifters of that detector have their own GUI configuration on the desktops. Depending on the tools available on the desktop, the users can access directly the cluster nodes via *ssh*.



Figure 38 ATLAS Control Room

Having only these two entry points in the ATLAS Online cluster, the access control is enabled on each entry point: at the Application Gateway level and Control Room Desktop level.

Figure 40 shows the cluster with its two entry points and a more in-depth look into the cluster organization from system administration point of view.

The cluster's nodes are grouped by TDAQ system components (see Figure 3) which are partitioned into sub-systems, typically associated with sub-detectors. Some nodes may be also shared between groups for various reasons (sharing data, common services for the clusters managed by experts from more areas). With this node grouping in place, it is obvious that users should be allowed to access only the nodes assigned to their groups and allowing them to perform the tasks relevant to their expertise domain.

Once the user is logged on a node where he is allowed to, he is free to work on that node or connect to another node from the cluster where he has access too. However, on each node there can be tools which require a higher level of protection and only a very specific expertise can use them. For this purpose, an additional level of access control is put in place to protect these sensitive tools on the nodes.

Figure 39 summarizes the access control enforcements at the system administration level. They are cascaded from the cluster entry points down to the specific tools on cluster nodes. This offers flexibility in choosing the right access control granularity on the cluster nodes: some of them are more important than others and only there it is suitable to make use of all access control mechanisms.

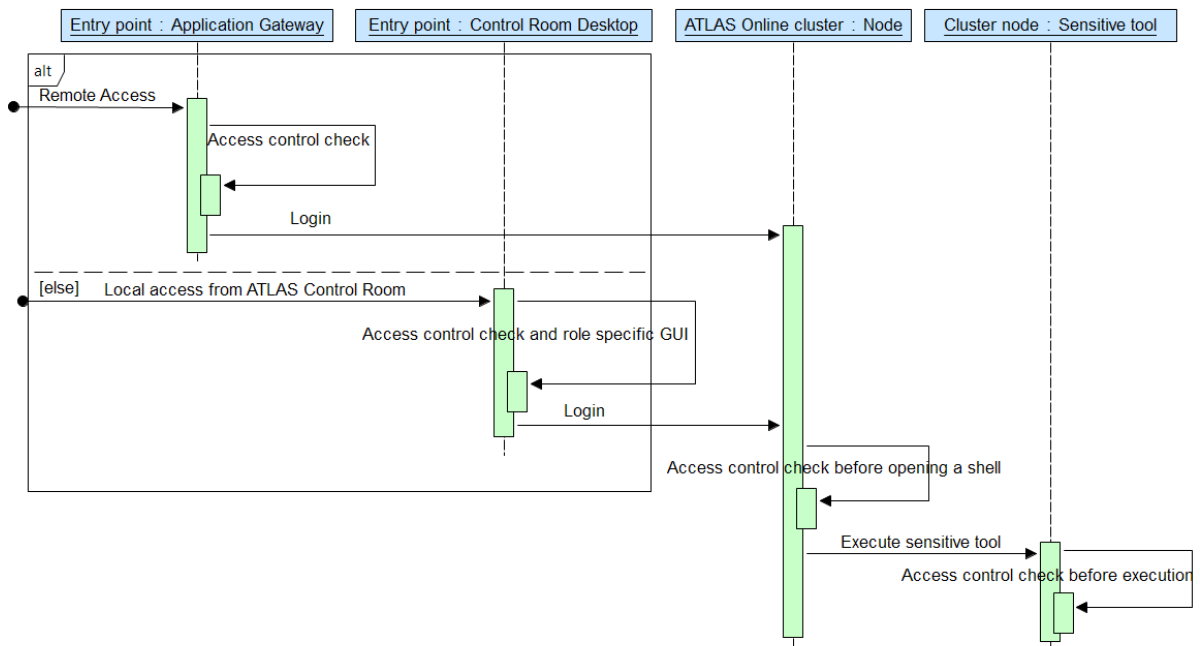


Figure 39 Sequence diagram of how users the ATLAS Online cluster.

The following sub chapters go into the details of each access control enforcement level.

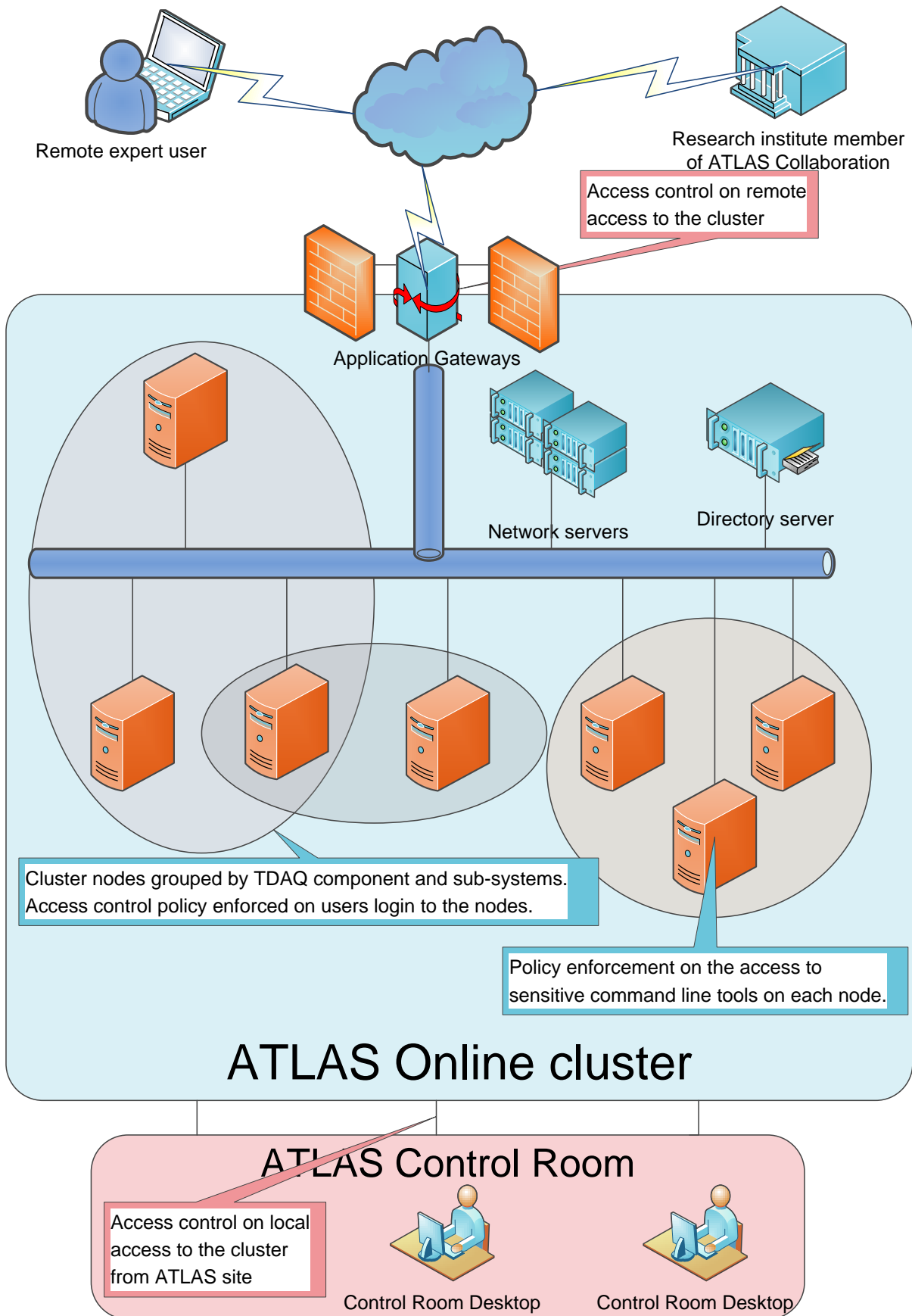


Figure 40 Access control enforcements at system administration level

5.1 Protection of entry points into the ATLAS Online cluster

The two entry points in the ATLAS Online cluster make use of the TDAQ Access Manager (AM) service (detailed later in chapter Chapter 6) to define the access control permissions and take the authorization decisions. The application running on each entry point uses the AM command line client to send authorization requests to the AM server and get the decision in return. It is then the client application responsibility to enforce the decision computed by the AM server.

The AM command line client is wrapped in a standalone script with the name `wrapped_amServerInterrogator`. Its help screen reads the following:

```
$ ./wrapped_amServerInterrogator -h
amServerInterrogator, version $Revision: 44692 $
Run the AM's Server Interrogator binary for the Operating System resource type.

Usage: amServerInterrogator [-s server_host] [-p server_port] [-l log_level] <-L
resource_location> <-P application> [-G arguments] <-A action> [-u
access_subject_username] [-o access_subject_hostname] [-h]

-s the hostname where Access Manager server is running
  Default is [pc-tdq-onl-ams]
-p the port number on which the Access Manager server is listening
  Default is [20000]
-l the ERS debug level. Should be a positive number
  Default is [0]
-L the resource location. Valid values: crd, gateway
-P the application to be accessed. Valid values: for crd-
>shell/lockscreen/am_tool, for gateway->login
-G the application arguments
-A the action to be performed. Valid values: for shell->open, lockscreen-
>lock/unlock, am_tool->role_state_change, for login->remote or inside
-u the access subject's username
  Default is []
-o the access subject's hostname
  Default is []
-h this info

The return codes are:
  0 - the authorization request has been successfully sent to the server and
the server granted the access to the resource
  1 - the authorization request has been successfully sent to the server and
the server denied the access to the resource
  any other number - the authorization request has not been sent to the
server or the request processing has failed
```

The usage of this tool is described under each Permission sub chapters below.

5.1.1 Remote access

The ATLAS Online cluster is not directly exposed to access from internet. The users logging in from internet must go first through the CERN LXPLUS service (`ssh username@lxplus.cern.ch`), then log in to the ATLAS Application Gateways (`ssh username@atlasgw.cern.ch`). Behind the *atlasgw* alias sits a battery of nodes running the Application Gateway: each of them accepts connections over *ssh* from CERN network and runs an application to assist the user in connecting further to a node inside the ATLAS Online cluster. One of the first steps performed by the application is to authorize the user logging in against the AM service. This authorization functionality can be turned on or off from a centralized configuration in LDAP server as shown in Figure 41. Turning off this authorization on the application gateways can be useful during maintenance periods or in case of a severe issue with the AM service which can block totally the remote access to the ATLAS Online cluster.

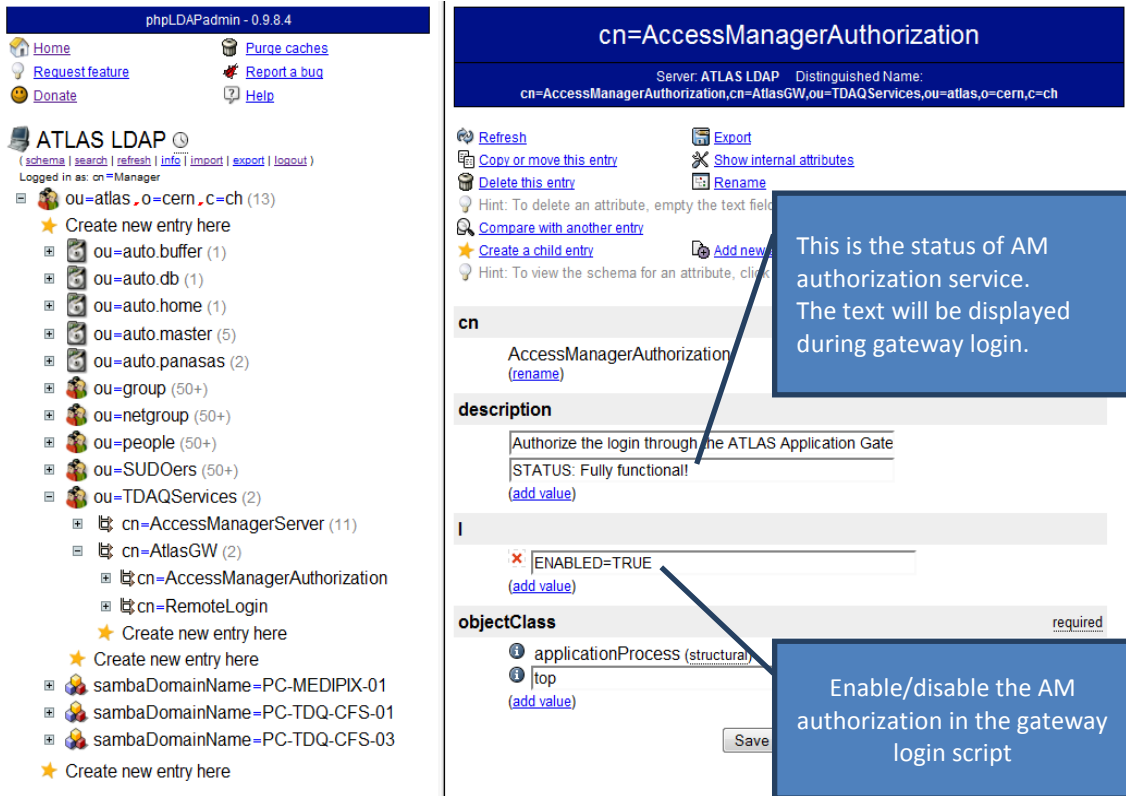


Figure 41 Centralized configuration of AM authorization for ATLAS Application Gateways

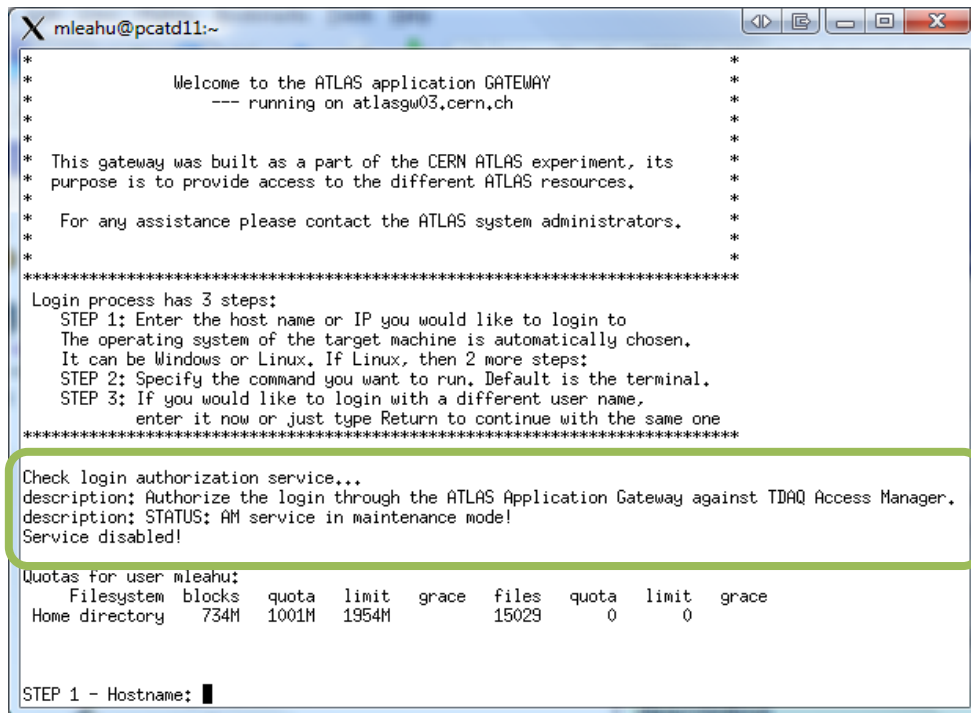


Figure 42 AM authorization disabled on the application gateway.

Figure 42 presents a screenshot of an attempt to access remotely a node in the cluster through the application gateway when the AM authorization is disabled. As it can be seen, the service status is *disabled* and the user is invited to provide the hostname of the destination node in the cluster.

```

mleahu@pcatd11:~
enter it now or just type Return to continue with the same one
*****
Check login authorization service...
description: Authorize the login through the ATLAS Application Gateway against TDAQ Access Manager.
description: STATUS: Fully functional!
Request authorization from TDAQ Access Manager service...DENIED(1)!!!
Please read the message below (5 seconds delay)...
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
YOU MUST HAVE AN EXPERT ROLE _ENABLED_ TO BE ABLE
TO ACCESS REMOTELY THE ATLAS ONLINE CLUSTER!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Please contact your system responsible to ask appropriate roles
BEFORE
reporting the problem to SysAdmins.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Should you think there is a problem with your Point 1 account or permissions,
please send an email to the ATLAS TDAQ SysAdmins with the following details:
----->cut here<-----
Date:Thu Oct 16 17:26:16 CEST 2008
User:mleahu
Roles assigned: [ATLAS:ShiftLeader ATLAS:observer DCS:shloperator TDAQ:CC:expert TDAQ:ROS:expert TDAQ:SBC:expert TDAQ:SYSADMIN:expert TDAQ:shifter TRT:shifter ]
Roles enabled: [ATLAS:observer ]
Gateway:atlasgw03.cern.ch
AM auth:[AUTHORIZATION=DENIED] [AMServerHistory=pc-tdq-onl-ams.cern.ch,][DECISION NotApplicable]
SSH_CLIENT:137.138.31.250 49906 22
SSH_CONNECTION:137.138.31.250 49906 137.138.19.186 22
----->cut here<-----
5 seconds delay...
Connection to atlasgw03-gpn closed.
[pcatd11] ~ $ █

```

Figure 43 AM authorization enabled on the application gateway.

The AM in action in the application gateway is shown in Figure 43:

- The AM authorization functionality is enabled in the application gateway
- The user trying to login has successfully passed the authentication phase, but the authorization has failed: *Request authorization from TDAQ Access Manager service ...DENIED(1)!!!*
- The user status shows that he has only the *ATLAS:observer* role enabled from the whole list of roles assigned. The reason for the authorization refuse is that the permission to access remotely the application gateway “is not applicable” to the set of role enabled for the user.
- The login procedure stops

The user found in this situation can obtain the access to the cluster by submitting a request to the shift leader to enable his expert role which inherits the remote access permission. Once the shift leader has enabled the role for the user, the next attempt to pass through the application gateway will be successful.

5.1.1.1 Permissions

The permissions and the association of permissions to roles are defined at the TDAQ AM service which is also in charge with the permissions management for its clients. More details on how the permissions are managed internally by AM are provided in its chapter (Chapter 6). Table 6 lists the permission details for the protection of the remote access to the ATLAS Online cluster. The permission is represented as an association of action that is allowed to be performed on a resource:

- The resource is characterized by three properties:
 - category: *os* showing that this resource is represented at the Operating System level being checked by the command line client
 - id: *gateway* because this is enforced at the gateway level
 - type: *login* refers to the gateway function
- There is only one action that can be performed on this single resource definition: *remote* access.

Table 6 Permissions for remote access

Resource			Action
<i>category</i>	<i>id</i>	<i>type</i>	
os	gateway	login	remote

This single permission definition is associated to the *RemoteAccess* role by the TDAQ AM service. This role is not assignable to users because it behaves as an intermediate role (see Figure 20) which collects all the low level permissions needed for someone to access remotely the ATLAS Online cluster. This role is then inherited by the assignable roles which need to have this high level permission: remote access to the online cluster.

The commands to prepare the *RemoteAccess* role, make the inheritance relationship with *TDAQ:expert* role and assign a user to the *TDAQ:expert* role are shown as an example below:

```
$ ./amRolesManager -c -r RemoteAccess
>>> Create NOT ASSIGNABLE role [RemoteAccess]...>>> DONE!

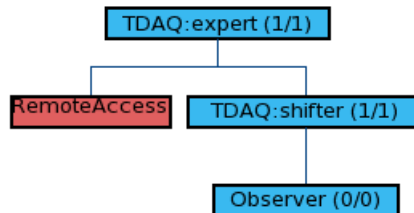
$ ./amRolesManager -s TDAQ:expert -r RemoteAccess
>>> add seniors for role [RemoteAccess]...
...senior [TDAQ:expert]:OK
modifying entry "cn=RA-RemoteAccess,ou=netgroup,ou=atlas,o=cern,c=ch"

modifying entry "cn=RE-RemoteAccess,ou=netgroup,ou=atlas,o=cern,c=ch"

>>> DONE!

$ ./amUserRoles -A TDAQ:expert -u mleahu
>>> Assign and enable roles to user [mleahu]...
>>> ...role [TDAQ:expert]...assign >>> OK! ...enable >>> OK!
```

Figure 44 shows the role hierarchy after the previous commands are run.

Figure 44 *RemoteAccess* role inherited by *TDAQ:expert*

Assuming that AM service is up and running, the remote access permission can be checked for the user set up in the example above:

```
$ ./wrapped_amServerInterrogator -s localhost -L gateway -P login -A remote -u
mleahu
$ echo $?
0
```

According to the tool return code (0), the authorization is successful and the authorization check logged on the AM server side reads:

```
[121020 15:54:39,541] ALLOWED Client[127.0.0.1:60701][access-subject(authn-
locality:dns-name=localhost.localdomain)(authn-locality:ip-
address=127.0.0.1)(subject-id=mleahu)intermediary-subject(authn-locality:dns-
name=localhost.localdomain)(authn-locality:ip-address=127.0.0.1)(subject-
id=mleahu)] requests access to [(resource-id=gateway)(resource-
type:application=login)(resource-category=os)] for [(action-id=remote)].
```

Disabling the role *TDAQ:expert* for the user, results in immediate authorization failure for the same permission:

```
$ ./amUserRoles -d TDAQ:expert -u mleahu
>>> Disable roles for user [mleahu]...
>>> ...role [TDAQ:expert]...disable >>> OK!
```

```
$ ./wrapped_amServerInterrogator -s localhost -L gateway -P login -A remote -u
mleahu
[AUTHORIZATION=DENIED]
$ echo $?
1
```

5.1.2 Control room desktops

The control room desktops (CRD) are installed with more KDE [46] desktop configurations which are chosen by the user at login time based. When the user logs in to a control room machine, the CRD's GUI initialization script retrieves the user's roles using `amUserRoles` script, then asks the user what role to use for the current session (Figure 45). Based on the role chosen by the user, the corresponding KDE configuration is loaded. The KDE GUI environment offered to the user is generated from KIOSK [47] profiles preinstalled on CRD, so the user has access only to the functions and applications specific to his current roles.



Figure 45 Choose roles when logging on CRD

Each KDE desktop configuration offers to the user applications which can be also under access control protection: shell, the possibility to lock or unlock the desktop screen, utility to manage user roles.

5.1.2.1 Permissions

As in the case of remote access permissions, the permissions used by the access control at CRD are managed by TDAQ AM service. Table 7 shows the permissions specific to CRD in terms of resources and their actions:

- The resources are of category *os* and id *crd*. There are three types:
 - *shell*: this represents a shell window opened in CRD
 - *lockscreen*: the screen locking feature
 - *am_tool*: the AM tool for RBAC model management
- The actions that can be performed on the resources are:
 - *open a shell*:
 - *lock or unlock the screen*
 - *change the user's role state from enabled/disabled (role_state_change) using the am_tool*

Hence, the valid checks with the AM command line tool are:

```
$. /wrapped_amServerInterrogator -s localhost -L crd -P shell -A open -u mleahu
$. /wrapped_amServerInterrogator -s localhost -L crd -P lockscreen -A lock -u mleahu
$. /wrapped_amServerInterrogator -s localhost -L crd -P lockscreen -A unlock -u
mleahu
$. /wrapped_amServerInterrogator -s localhost -L crd -P am_tool -A role_state_change
-u mleahu
```

Table 7 Permissions for CRD

Resource			Action
category	id	type	
os	crd	shell	open
		lockscreen	lock
			unlock
		am_tool	role_state_change

5.2 Protection on cluster nodes

The protection mechanisms on the cluster nodes are implemented using the standard Linux features without the need of interaction with the TDAQ AM service. The RBAC data (users, roles, users association to roles and roles hierarchies) are shared between these mechanisms and the TDAQ AM service. The following chapters detail the login restrictions enforcement on cluster nodes, then the protection of the sensitive tools on the cluster nodes.

5.2.1 Login restrictions

This access control mechanism makes sure that users are allowed to log in (either locally from the computer's console or remotely via *ssh*) only on the machines which are allocated to groups they belong.

In order to make a fresh SLC 5 node aware of centralized LDAP configuration, its local services must be adjusted to look up the LDAP server to read their configuration and other information they may need. The LDAP support is enabled on the "Authentication Configuration" (as shown in Figure 46) for "User Information", "Authentication" and other Options (especially the authorization based on `access.conf`). After this change, the following configuration files are updated:

- the LDAP configuration file `/etc/ldap.conf` used by all tools from the current node that need to know the identifier of central LDAP server. Its content can be like the following one:

```
# the server hostname where LDAP service is installed
host localhost

# the Base Distinguish Name to use on the LDAP service
base ou=atlas,o=cern,c=ch
```

- the PAM generic configuration `/etc/pam.d/system-auth` includes calls to `pam_ldap.so` plugin in all its stacks (*auth*, *account*, *password*, *session*). Consequently, all system services (e.g. *sshd*, *sudo*, *login*) making use of PAM stack for their user authentication, account information, user password management and user session management will access the LDAP server as a central information repository. The `pam_access.so` is also added in the *account* stack to enforce the authorization rules defined in the `/etc/security/access.conf`. The *account* stack definition looks like this:

```
account required pam_access.so fieldsep=|
account required pam_unix.so broken_shadow
account sufficient pam_succeed_if.so uid < 500 quiet
account[default=bad success=ok user_unknown=ignore] pam_ldap.so
account required pam_permit.so
```

- the Name Service Switch [48] configuration file `/etc/nsswitch.conf` has LDAP as second source of information for system services. For example, at least the following services will lookup LDAP:

```
passwd:      files ldap
shadow:     files ldap
group:      files ldap
netgroup:   ldap
automount:  files ldap
sudoers:    files ldap
```

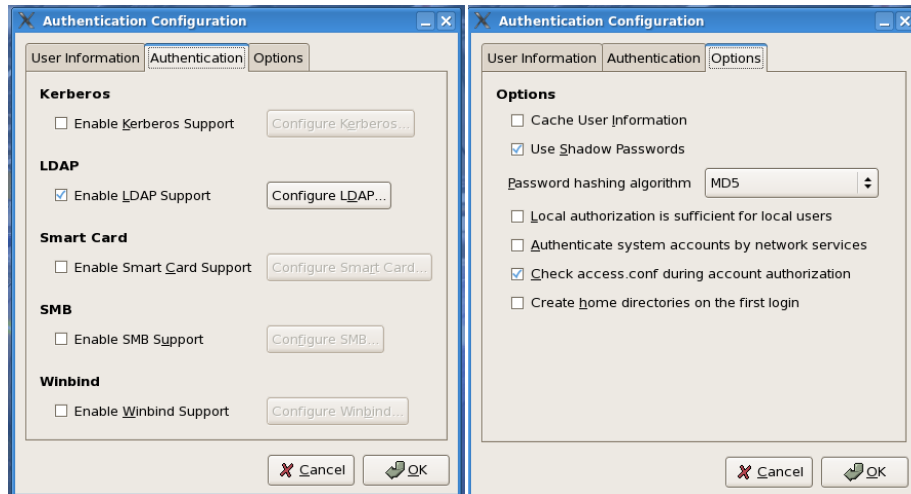


Figure 46 “Authentication Configuration” in SLC 5

At this stage, the node is ready to enforce login restrictions based on the permissions defined in `/etc/security/access.conf` [49].

5.2.1.1 Permissions

The permissions and permission assignment to role from our RBAC model are centralized in LDAP in the form of sudo roles. The example in Figure 47 shows the main characteristics of a sudo role definition that transforms it in a login restriction definition:

- sudo role name which represents the *permission name* must start with `LOGIN-` keyword.
- permission* is defined as the hostname where the login is allowed. The hostname can be also a simple regular expression (for example, all the nodes with the hostname starting with `pc-tdaq-control-`), thus making the permission valid for a group of machines obeying a hostname naming convention (e.g. the hostname can have the structure `<type of machine: pc/sbc>-<group name>-<subgroup>-<index>`).
- permission assignment to role* is given by the netgroup name specified in the `sudoUser` field. This netgroup must correspond to a role defined in LDAP as mentioned previously in chapter 4.2.
- the other attributes of a sudo role in LDAP are set to default values as in the example below.

The permission name: LOGIN-*

The permission

The permission assignment to role

LDAP configuration details:

- cn: LOGIN-RE-TDAQ:shifter (required, rdn)
- description: Login access control permission for TDAQ:shifter (add value)
- objectClass: sudoRole (structural), top (required) (add value)
- sudoCommand: /bin/lis (add value)
- sudoHost: pc-tdaq-control-* (add value)
- sudoOption: !authenticate (add value)
- sudoRunAs: nobody (add value)
- sudoUser: +RE-TDAQ:shifter (add value)

Figure 47 Example of login restriction in LDAP

The `access.conf` file is generated by a shell script included in the appendix 8.6. Its help screen reads:

```
$ ./amLoginRestriction -h
amLoginRestriction - $Revision: 1.18 $
Get the AM specific rules from LDAP to restrict the login on cluster nodes based on
roles/netgroups.
Usage: amLoginRestriction [-n machine_name] [-u] [-B] [-s] [-c minutes] [-C] [-r]
[-l ldapserver] [-H ldapuri] [-b basedn] [-v] [-h]
  -n specify the machine where the rule applies. Default is the current machine.
  -u update the file [/etc/security/access.conf]
  -B make a backup copy of [/etc/security/access.conf] before update
  -s show the users allowed to login to the machine_name as specified in LDAP rules
  -c set a cronjob in [/var/spool/cron/mleahu] on the current machine to run the
command [/home/mleahu/am_scripts/amLoginRestriction -u] every 'minutes' minutes
  -C remove the cronjob set with -c if any found. If -c provided in the same time,
then first the current cronjob is removed.
  -r restart the services with PAM support (e.g., sshd)
  -l use this ldapserver; default is [localhost]
  -H use this ldapuri; default is []
  -b use this basedn; default is [ou=atlas,o=cern,c=ch]
  -v verbose mode
  -h this info

Author: mleahu@CERN
```

This script is set to run periodically on the node as a cron [50] job to look up in LDAP all permissions defined for the current node. An example of its execution output is shown below:

```
# ./amLoginRestriction -n pc-tdaq-control-001 -v
>>> ===== LOGIN RESCTRITION CONFIGURATION =====
```

```
ldapsearch -h localhost -b ou=atlas,o=cern,c=ch -x -LLL -S cn (&(|(sudoHost=pc-tdaq-control-001)(|(sudoHost=pc-tdaq-control-\*)(|(sudoHost=pc-tdaq-\*)(sudoHost=pc-\*))))(&(cn=LOGIN-*)(objectClass=sudoRole)) sudoUser cn sudoHost
dn: cn=LOGIN-RE-TDAQ:shifter,ou=SUDOers,ou=atlas,o=cern,c=ch
cn: LOGIN-RE-TDAQ:shifter
sudoHost: pc-tdaq-control-*
sudoUser: +RE-TDAQ:shifter

>>> PAM ACCESS configuration generated from LDAP information!
#
# Login access control table.
# Generated automatically by the script './amLoginRestriction'
#
# SUDO RULE DN:
# cn=LOGIN-RE-TDAQ:shifter,ou=SUDOers,ou=atlas,o=cern,c=ch
+ | @RE-TDAQ:shifter | ALL
- | ALL EXCEPT root | ALL
```

We can check also the users who get the access to this node and it can be observed that alice is also granted the permission through the permission inheritance over role hierarchy:

```
# ./amLoginRestriction -n pc-tdaq-control-001 -s
>>> Users allowed to login to [pc-tdaq-control-001]:
>>> Netgroup [RE-TDAQ:shifter]:
alice
mleahu
```

5.2.2 Access control to sensitive tools

This mechanism comes on top of the previous one to increase the granularity of access control to the sensitive tools available on the node.

The prerequisite of enabling this level of access control is the presence (by default in SLC 5) of *sudo* [30] application on each node of the Linux network. The application description provided in [30] states: “*Sudo (su "do") allows a system administrator to delegate authority to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while providing an audit trail of the commands and their arguments*”. Since we are interested in having a centralized LDAP configuration of this access control mechanism, the *sudo* tool must be available with support for LDAP – by default in SLC 5. The *sudo* configuration must be also adjusted to point to the right LDAP server and base DN:

- `/etc/ldap.conf` must contain the Base DN where the sudo roles are defined besides the server identifiers already mentioned in the previous chapter:

```
sudoers_base ou=SUDOers,ou=atlas,o=cern,c=ch
```

5.2.2.1 Permissions

The protection of tools (e.g. shell scripts or binaries) subject to access control must be done in two steps:

- restrict the file access permissions of the targeted tool to a generic Linux user created only for this purpose. For example, the tool `FarmToolsLauncher` is owned and allowed to be run only by `farmtoolsuser`:

```
chown farmtoolsuser /sw/tdaq/scripts/FarmToolsLauncher
chmod u=rwx /sw/tdaq/scripts/FarmToolsLauncher
chmod og-rwx /sw/tdaq/scripts/FarmToolsLauncher
```

- prepare a sudo role in the LDAP to allow the execution of the protected tool. Figure 48 shows an example of sudo role for the tool `FarmToolsLauncher` where:
 - the *permission name* is by convention the sudo role name
 - the *permission* is represented by the tool identifier and the user to run as
 - the *permission assignment to role* is defined as the netgroup corresponding to the role allowed to execute the tool

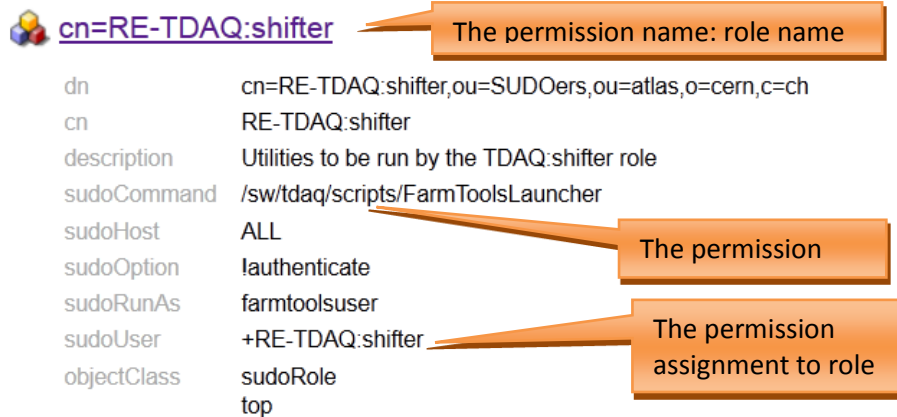


Figure 48 Example of sudo role in LDAP

Assuming that role hierarchy from Figure 28 where the user `alice` has the role `ShiftLeader` assigned and enabled, then we can check her `sudo` permissions by running the following `sudo` command:

```
# sudo -l -U alice
User alice may run the following commands on this host:
(farmtoolsuser) NOPASSWD: /sw/tdaq/scripts/FarmToolsLauncher
```

Hence, the `alice` user is able to run the `FarmToolsLauncher` application thanks to the permission defined in LDAP and permission inheritance over the role hierarchy. Disabling `alice`'s role makes her `sudo` permission to be revoked:

```
# ./amUserRoles -d ShiftLeader -u alice
>>> Disable roles for user [alice]...
>>> ...role [ShiftLeader]...disable >>> OK!
# sudo -l -U alice
User alice may run the following commands on this host:
```


Chapter 6

Access control at the TDAQ Online Software level

The access control at the TDAQ software level is implemented with the help of a dedicated service: TDAQ Access Manager Service (referred from now on as AM). The need for this dedicated service was first acknowledged in “ATLAS high-level trigger, data-acquisition and controls: Technical Design Report” [10] where it is seen as a component of the Online Software – Control package with the following high level scope:

Access Management: *The control package provides a general Online Software safety service, responsible for TDAQ user authentication and the implementation of an access policy for preventing non-authorized users corrupting TDAQ functionality.*

The formal requirements for AM were drafted in document “ATLAS TDAQ Online Software – Access Manager Requirements” [51] and the first implementation followed the design referenced in [52]. This implementation was tightly integrated in the TDAQ Online Software infrastructure where it was intercepting the communication (Inter Process Communication - IPC) between TDAQ components to authorize the requests made by a TDAQ component to another one. The RBAC data was stored in the AM internal relational database. After the experience gathered with this implementation, it was understood the need for a wider scope of AM service:

- The access control must be applied not only in the components of TDAQ Online Software, but in the other applications running in the ATLAS Online cluster. This requires that RBAC model is well “understood” by all entities involved, even by the operating system specific protection mechanisms.
- The enforcement mechanism should not be dependent on the communication between TDAQ components because the communication can take various forms (not only IPC) and the solution should scale independent of communication channels.
- The non-functional requirements (high availability, performance) are important for this service and are not trivial to fulfill.

We took the challenge of extending the AM solution and implement the AM service to fulfill the high level requirements mentioned above. The AM service has kept the same client-server approach as in the first implementation. We designed the Access Manager server as an highly scalable server architecture based on the reactor pattern to handle hundreds of client requests in parallel. It retrieves the RBAC data from the databases, listens for authorization requests from clients, processes the requests with the access policies taking into account the user’s roles enabled at that time, and communicates the response to the client.

The following chapters describe the requirements for the AM service, how it is designed and important aspects from its implementation. The tests run on the test bed are described from two perspectives: functional tests (both unitary and integration tests) and performance and stress tests to prepare the production setup. The production setup in the ATLAS Online cluster is presented extensively in a dedicated chapter outlining the deployment of a high available cluster of AM servers, notification mechanisms from LDAP servers and integration with the SysAdmin monitoring tool.

6.1 Requirements

Following the experience with the first AM implementation, the requirements have changed towards a more open scope of AM in the ATLAS Online cluster. The main changes in the high level requirements are:

- The goal is to have an AM service open for integration with more client types besides the TDAQ Online software components. The remote access protection and access control in CRD are a few examples of such integrations.
- The main RBAC model entities (users, roles and their relationships) are not anymore in the scope of the TDAQ AM service, but at a higher level to be usable by other enforcements mechanism such as the OS specific ones detailed in the previous chapters. The RBAC Administration Tool is in charge with that, and only the permissions management remains in the scope of AM service. As a future action point is the integration of RBAC Administration tool with all the permissions management entities.
- The AM service should not be intrusive in the way its clients communicate between each other.
- The AM service takes the authorization decision. Its enforcement is in charge of the client who requested the authorization decision.

6.1.1 Assumptions

The AM service must run in the ATLAS Online cluster and be open for interrogation from every node of this cluster. This implies that the communication between clients and AM service is directly through the network without any intermediate entities such as Network Address Translation service, Application Gateway, Load Balancers.

The ATLAS Online cluster is assumed to be isolated from the outside world (CERN General Public Network, Public Internet) and a certain level of trust can be put on this environment. This implies that secured communication between clients and AM service is not critical or mandatory.

The authentication of user on clients using AM Service is handled by cluster system administration services (the OS specific mechanisms) and is not to be addressed by the AM Service.

The hardware configuration of nodes running the AM Service is a standard server specification for ATLAS Online cluster.

6.1.2 Functional requirements

The main use cases (Figure 49) to be implemented by the AM service are the following:

- *Permission Management*

The permissions definition is composed of resource types and their actions definition. These are tightly coupled with the algorithm for taking the authorization decision; hence they must be addressed in the context of AM Service.

- *Permission association to Roles*

Since the permissions are defined by AM Service, the association of permissions to roles must be performed here too. These permissions and permissions association relationships are the last pieces of the RBAC model deployment in ATLAS Online cluster – the TDAQ service part (the system administration level was addressed in the previous chapters).

- *Check authorization for actions on system resources*

The AM Service must answer to authorization requests from clients based. The authorization request must provide all necessary information to take a decision based on the

policies configured in the AM Service. This authorization request information contains the following mandatory data:

- *Resource identifier*: the unique identifier of the resource to be accessed
- *Action*: the action to be performed on the resource
- *The user to act on the resource*: the identifier of the user who will act on the resource if the authorization is granted. Usually, a generic software application acts on behalf of other users, so the one who asks for authorization must ensure that the user on behalf of who is running is allowed to perform the action on the resource. Of course, there can be simple cases when the user asking for authorization will also perform the action on the resource.

The AM service must answer to the client with a decision (authorization granted or denied) and, when possible, with a reason which is useful for the client in cases of denial. Besides taking authorization decisions on client requests, the AM Service must record each authorization requests for later audit purposes.

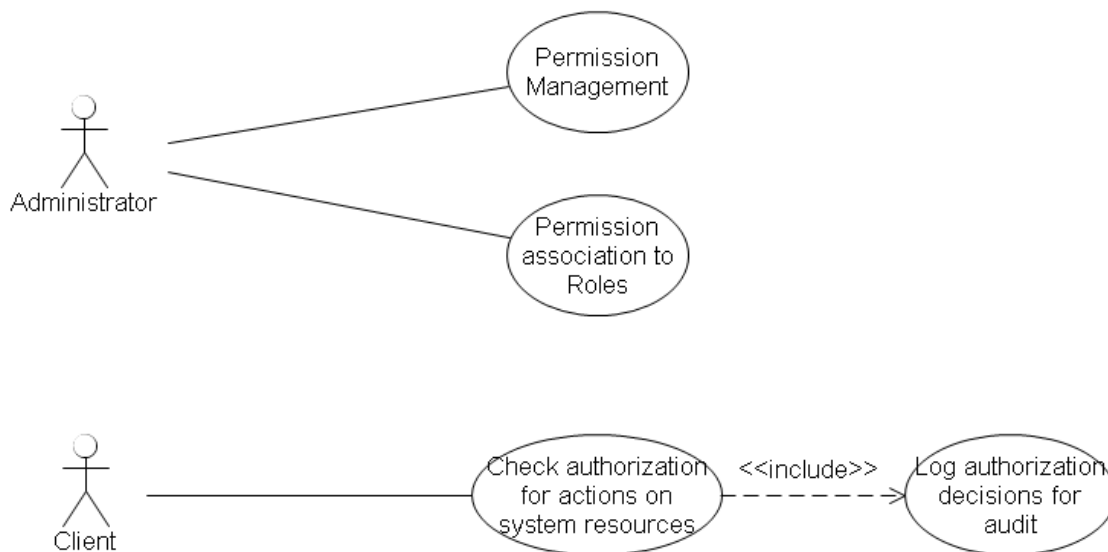


Figure 49 TDAQ AM Service use cases

The actors can be of two types:

- *Administrator*: The human user who is responsible for defining the access control policies for the TDAQ components integrated with the AM service.
- *Client*: Software application that performs actions on resource on behalf of other users (human users or other applications).

6.1.3 Non-functional requirements

The most important non-functional requirements for this service operation consist in:

- *Availability*: A system's availability, or "uptime," is the amount of time that it is operational and available for use. The AM Service must run without interruption as long as the experiment is in the running state too. However, in case there is a major unscheduled downtime of this service, there must be put in place a procedure to control the access control enforcements in ATLAS Online cluster to not block the experiment functioning.
- *Flexibility*: The AM service must be designed in such way to allow future integration in the access control of TDAQ components or other services in the ATLAS Online cluster which are not included in the scope at this moment.

- *Portability*: The AM service must be able to run on current and future versions of SLC since this is the official version of Linux to be used in the ATLAS Online cluster.
- *Performance*: The clients which integrate access control checks in their functioning should not be impacted from performance point of view by the interaction with the AM service. A measurable requirement for this aspect is to have the time spent on the server to process an authorization request of the order of hundreds of milliseconds at most.
- *Reliability*: Reliability specifies the capability of the software to maintain its performance over time. The AM service must run continuously with 0 failures during experiment functioning.
- *Robustness*: A robust system is able to handle error conditions gracefully, without failure. This includes a tolerance of invalid data, software defects, and unexpected operating conditions. The AM service must be designed in such way to handle unexpected errors that may occur on the environment where it runs.
- *Scalability*: Since the ATLAS Online cluster is likely to increase its size, the AM service must be designed to scale horizontally (scale out).

6.2 Design

Challenges to design access control software systems were taken in the past by many governmental agencies and enterprises, each one in their own proprietary manners in setting up the access control model and communication protocols between system entities. These differences made the integration of various access control systems difficult and inter-operability required many translation layers. At the same time, there is increasing pressure on corporate and government executives from consumers, shareholders and regulators to demonstrate "best practice" in the protection of the information assets of the enterprise and its customers. Hence the need for standardization emerged.

XACML (eXtensible Access Control Markup Language) [13] is an initiative of OASIS (Organization for the Advancement of Structured Information Standards) [32] to develop a standard for access control and authorization systems. The standard defines a declarative access control policy language implemented in XML (eXtensible Markup Language) [53] and a processing model describing how to evaluate authorization requests according to the rules defined in policies.

As a published standard specification, one of the goals of XACML is to promote common terminology and interoperability between authorization implementations by multiple vendors. XACML is primarily an Attribute Based Access Control system (ABAC), where attributes (bits of data) associated with a user or action or resource are inputs into the decision of whether a given user may access a given resource in a particular way. Role-based access control (RBAC) can also be implemented in XACML as a specialization of ABAC.

After analyzing the AM service needs and the XACML standard, we found XACML to be suitable for building RBAC with it in the AM service. Hence the high level design of AM service follows the XACML approach outlined in the XACML data flow model in Figure 50. The XACML model operates in the following steps (as described in [54]):

1. PAPs write policies and policy sets and make them available to the PDP. These policies or policy sets represent the complete policy for a specified target.
2. The access requester sends a request for access to the PEP.
3. The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource, action and environment.
4. The context handler constructs an XACML request context and sends it to the PDP.
5. The PDP requests any additional subject, resource, action and environment attributes from the context handler.
6. The context handler requests the attributes from a PIP.

7. The PIP obtains the requested attributes.
8. The PIP returns the requested attributes to the context handler.
9. Optionally, the context handler includes the resource in the context.
10. The context handler sends the requested attributes and (optionally) the resource to the PDP. The PDP evaluates the policy.
11. The PDP returns the response context (including the authorization decision) to the context handler.
12. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP.
13. The PEP fulfills the obligations.
14. (Not shown) If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

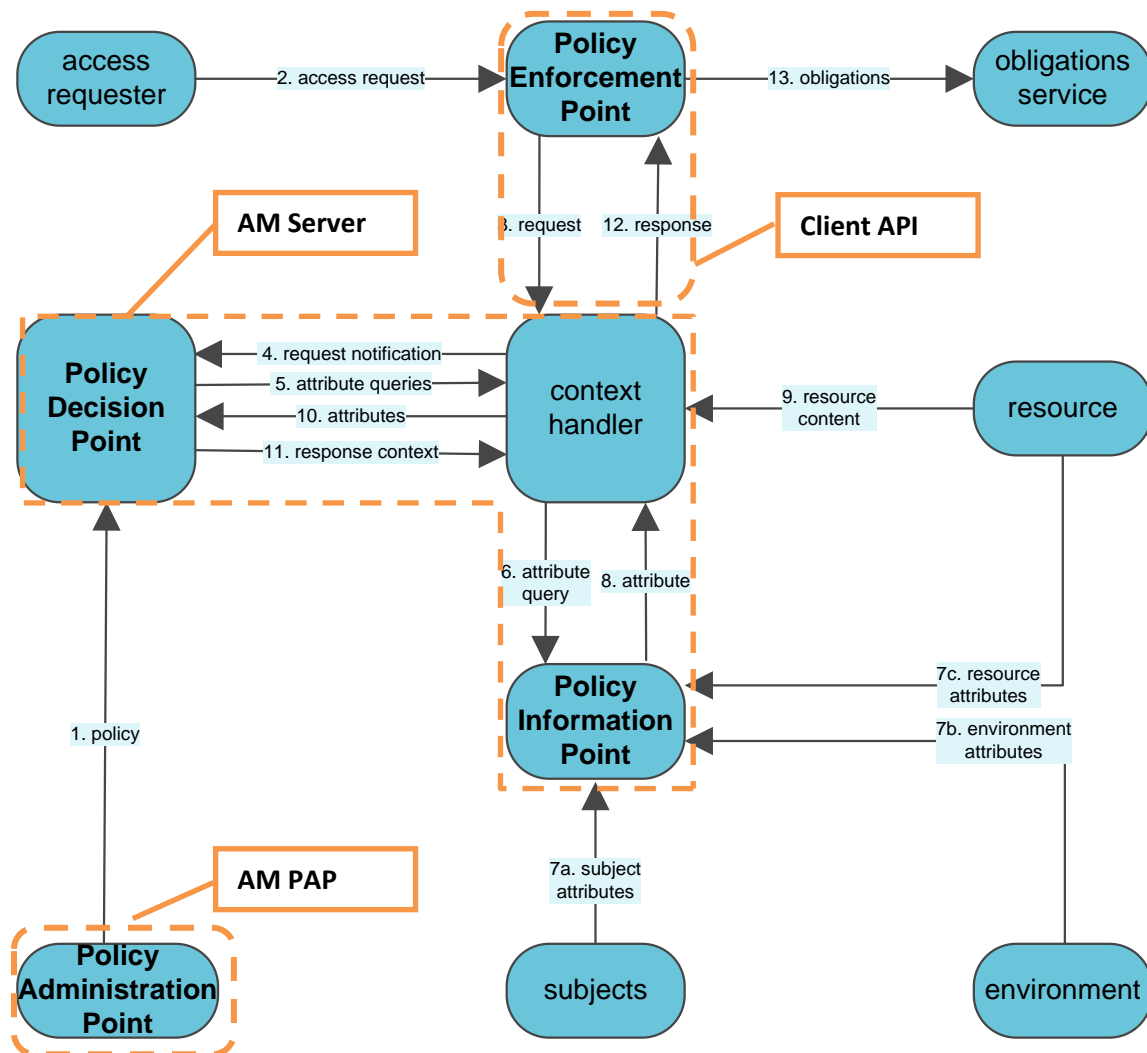


Figure 50 Mapping of XACML actors to AM service components

The mapping of XACML actors to AM service components is the following:

- *Policy Administration Point (PAP)*: This component of AM service corresponds to PAP from XACML and addresses the use cases *Permission Management* and *Permission association to Roles*. It prepares the policies in XACML format using the RBAC concepts.

- AM server and APIs: These client-server components implement the use case *Check authorization for actions on system resources*. The AM server corresponds to XACML’s PDP, PIP and the client API to the PEP.

The following chapters detail the design of AM service components ending with a summary of requirements coverage by the design.

6.2.1 Policy Administration Point

This component’s responsibility is to translate the permissions defined in “ATLAS language” into policies in XACML format to be used by the AM server when taking authorization decisions.

The **input** of this component is represented by the permissions specified in a simple format with the resources and actions specific to ATLAS needs (the sysadmin or TDAQ components specific permissions). The permissions and permissions assignment to roles are expressed in an input file with the following format:

```
# The lines starting with # are comments

##### PERMISSIONS DEFINITIONS AS RULES #####
# The rules should be in the following format:
# [Rule=_the_rule_name_] [ResourceCategory=<value>] [ResourceId=<value>]
[ResourceType=] [<custom_property_specific_to_some_resource_types>=<value>]
[ActionId=<value>]

##### RULES ASSIGNMENTS TO ROLES #####
#
# [Role=_the_role_name_] [IncludeRule=_rule_1] [IncludeRule=_rule_2]
#
```

The components of a *permission* rule are summarized in Table 8. The values of each component can be a strict value (a string) or a regular expression (in this case, the value must start with {*regexp*}).

Table 8 The components of a permission from PAP input file

Permission Rule Component	Component Property	Description
Resource	ResourceCategory	The high level category of the resource. This can correspond to a TDAQ component (e.g. <i>PMG</i> – Process Manager), a sub system (e.g. <i>BCM</i>) or another area of resources (e.g. <i>os</i> - Operating System specific tools and tasks). This is mandatory.
	ResourceId	These play the role of sub category of ResourceCategory and are specific to each category instance.
	ResourceType*	
Action	ActionId	The identifier of the action to be performed on the resource.
Decision	<i>Permit, Deny</i>	By default, a rule has the meaning of “Permit the action on the resource”. It is possible to change the rule purpose by specifying another decision to be considered if an authorization requested matches this rule.

A rule can consist in one or more associations of resources, actions and decisions, meaning that more lines in the input file can have the same rule name. For example:

```
[Rule=rule1] [ResourceCategory=category1] [ResourceId=id1] [ActionId=action1]
[Rule=rule1] [ResourceCategory=category11] [ResourceId=id11] [ActionId=action11]
```

In this case, the rule is matched against an authorization request if the resource and actions from the request match one of the resource and action associations defined for the rule.

The following tables list the permissions rule components for various resource categories specific to TDAQ components.

Table 9 Process Manager resource category

Resource					Action
Resource Category	ResourceId	ResourceType Hostname	ResourceType Arguments	ResourceType OwnedByRequester	ActionId
pmg	<process_binary_path>	<hostname>	<arguments>	true/false	start
					terminate

Table 10 Run Control resource category

Resource			Action
Resource Category	ResourceType Command	ResourceType Partition	ActionId
RunControl	<empty> (meaning any command)	<partition_name>	exec_cmd
	publish		
	publish_statistics		

Table 11 IGUI resource category

Resource		Action
ResourceCategory	ResourceId	ActionId
IGUI	display	view
	control	
	expert	

Table 12 Resource Manager resource category

Resource		Action
ResourceCategory	ResourceTypePartition	ActionId
ResourceManager	<partition_name>	free
		lock

Table 13 Data Base resource category

Resource			Action
ResourceCategory	ResourceId	ResourceTypePath	ActionId
DataBase	directory	<directory_path_value>	create_subdir
			delete_subdir
	file	<file_path_value>	create_file
			update_file
			delete_file
	admin		admin

Table 14 BCM resource category

Resource		Action
ResourceCategory	ResourceId	ActionId
BCM	bcm	configure
		update

The *permissions assignment to roles* consists in associating permissions rules to the role names. These roles must be already defined in the central directory server where roles and roles hierarchies are stored. For example, a role can have one or more permissions associated:

```
[Role=TDAQ:expert] [IncludeRule=rule1] [IncludeRule=rule2]
```

The **output** of PAP component is represented by the set of XACML policy files that represent the permissions in the form of XACML rules and their association to roles and roles hierarchies. The following chapter details the XACML policy structure.

6.2.1.1 XACML language for policy storage

The policy language model as defined [54] is shown in Figure 51 and consists in the following components: Rule, Policy and Policy set.

A **rule** is the elementary unit of a policy and can not be exchanged between actors unless it is encapsulated in a policy. The evaluation of a rule is based on its content:

- **target** composed of **subject**, **resource**, **action** and **environment**. The meaning of the target is that the subject wants to perform the action on the resource in a given environment.
- **effect** indicates what is the purpose of the rule in case of matching the request with the target. The effect can be "Permit" or "Deny".
- **condition** represents a Boolean expression that refines the applicability of the rule beyond the predicates implied by its target. Therefore, it may be absent and we don't use it.

A **policy** aggregates more rules which are evaluated and the final decision for this policy is taken by a "rule-combining-algorithm". The target of the policy has the same structure as in the case of rules. The obligations defined for a policy are returned to the clients if their request matched this policy. We don't use obligations in our solutions.

A **policy set** gathers more policies and is similar to a policy component. One important difference is that a policy set can reference another policy set which allows the modeling of a policy set hierarchy.

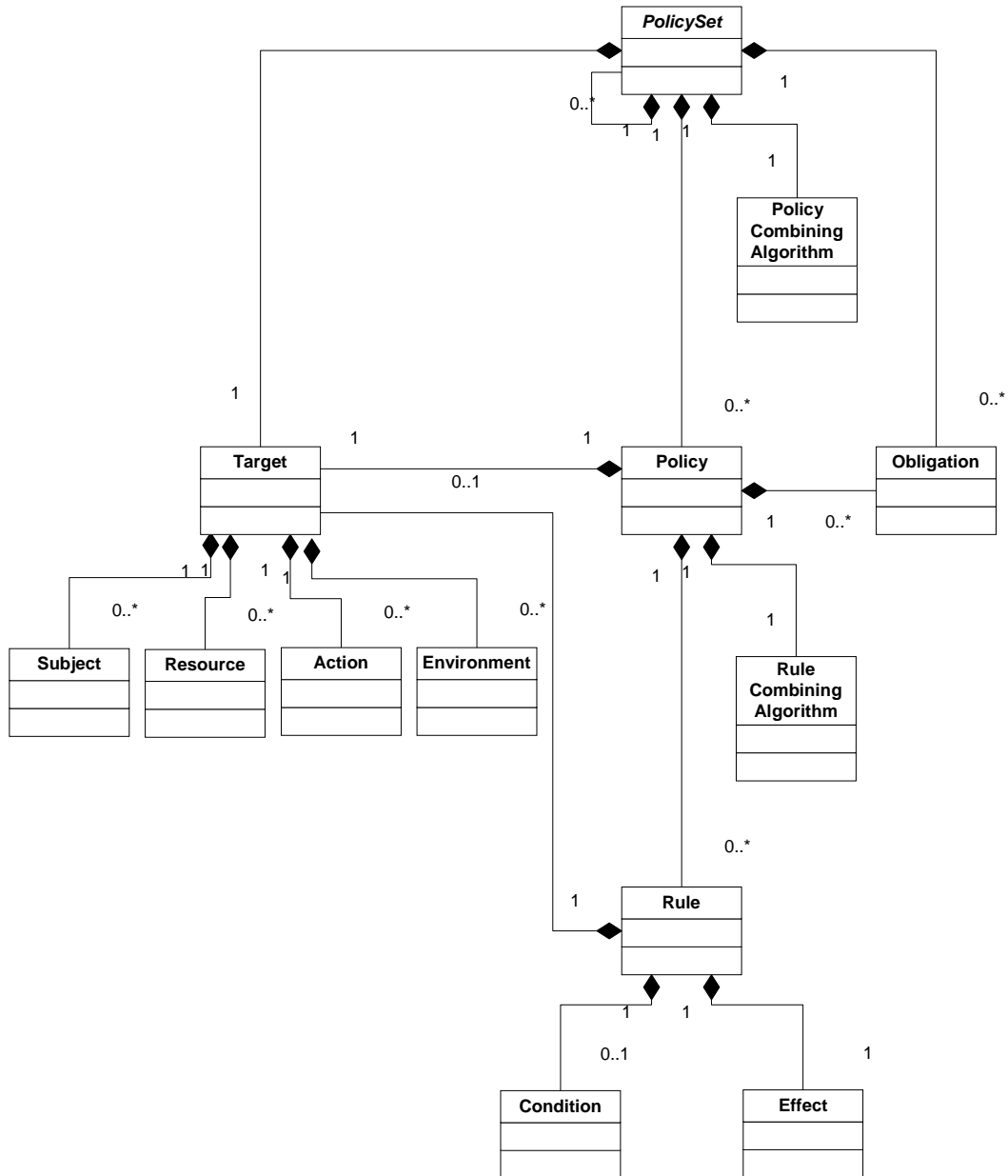


Figure 51 XACML policy language model

The Hierarchical RBAC policy model used in the TDAQ AM service is a version of the generic XACML policy model shown in Figure 52 and based on the recommendations from [55]. Its main characteristics are the following:

- the rules have a target composed of resource types and actions without subject and environment. The rule’s default effect is “Permit” and no condition is attached.
- the rules for resources from the same category and same id are gathered in a single **Permission Policy (pp)**. The target of a **pp** has defined only the resource id attribute. The rule combining algorithm is “rule-combining-algorithm:deny-overrides”. No additional obligations are set.

- **Permissions Policy Set (ppsrule)** gather more **Permission Policies** which contain rules for resources of the same category. The target of ppsrule contains only the resource category identifier. The ppsrule references the other pp by <PolicyIdReference> tag. The policy combining algorithm is “policy-combining-algorithm:first-applicable”.
- From the roles perspective, we use two levels of role policy sets:
 - The top ones are the **Role Policy Sets (rps)** which have the target composed only of the subject role attribute. The rps references further the second level of policy sets.
 - The second level is represented by **Permission Policy Set for Roles (ppсроles)**: the policy sets used only to link roles to permissions (**ppsrules**) AND implement the roles hierarchies (references other **ppсроles**). The ppsroles don’t have any target, just <PolicyIdReference> tags.

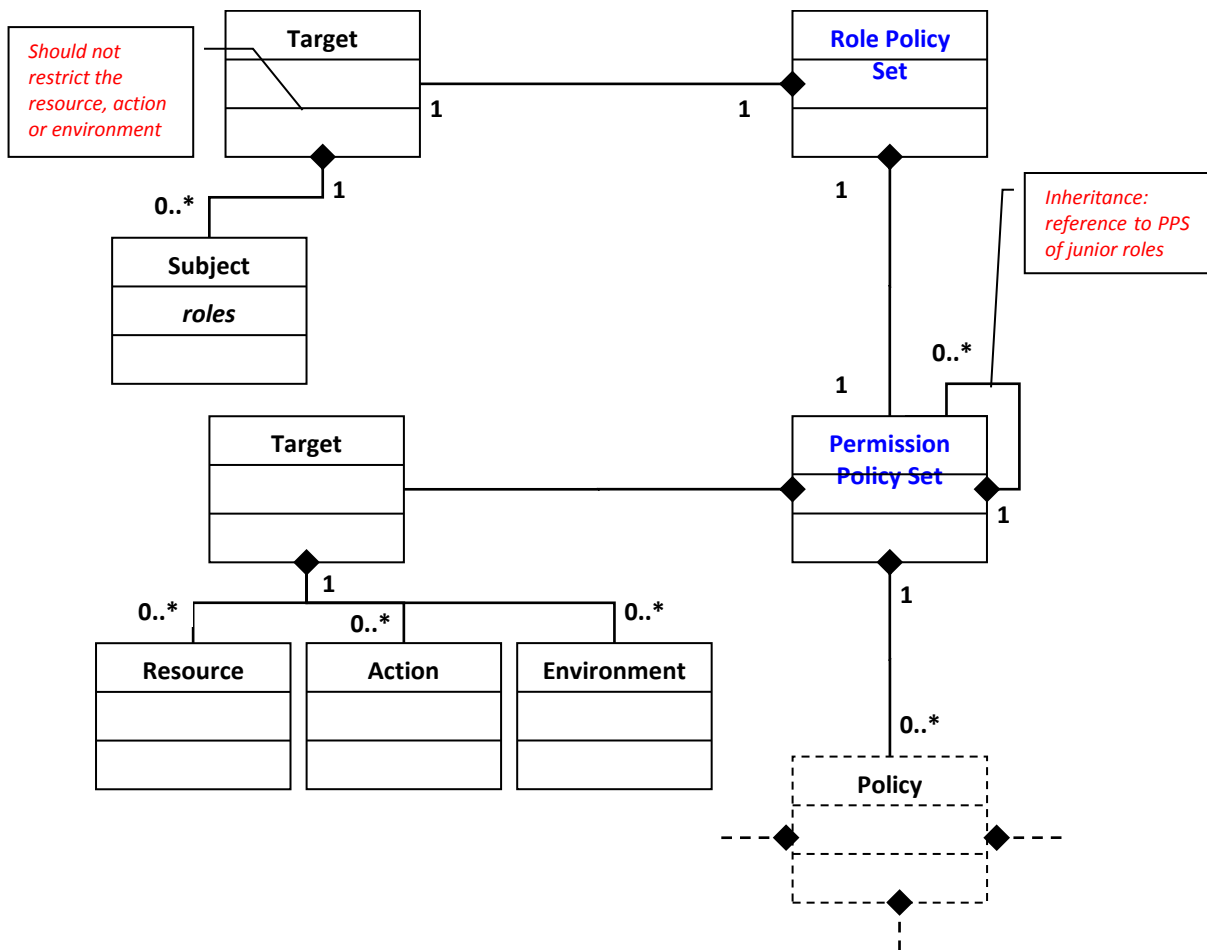


Figure 52 Hierarchical RBAC profile of XACML

6.2.2 Client-server model

The XACML model supports and encourages the separation of the authorization decision from the point of use. When authorization decisions are taken into client applications (or based on local machine user identifiers and Access Control Lists), it is very difficult to update the decision criteria when the governing policy changes. When the client is decoupled from the authorization decision, authorization policies can be updated on the fly and affect all clients immediately.

We’ve designed the AM service on the client server model with the client asking the server for authorization decision, the server taking the decision based on the RBAC data and replying to the

client with the decision result. It is the client then who stops the user action or not depending on the authorization decision.

The following chapters describe the AM server high level design and the client API to be used by the server's clients. At the end the critical non-functional aspects of AM service are addressed.

6.2.2.1 Server

The server component has the main responsibility to handle authorization requests from clients by taking a decision and communicating this decision back to the requester. The UML [56] component diagram in Figure 53 shows the high level interfaces exposed by the server and the interaction with other XACML specific components. The main functional interfaces are the following:

- **Authorization requests listener** receives network connections from clients, reads the request content and pass it further to other internal components for processing. Once the request processing has finished, the response is sent back to the client over the same network connection.
- **Controller interface** is meant to be used for monitoring of AM server functioning. Server information and its Key Performance Indicators (KPI) are exposed over this interface:
 - Server start timestamp
 - Server up time
 - Various server counters about the number of authorization requests handled from service start up
 - The average values of server counters
 - The peak values of server counters

Short notifications can be also received over this interface (e.g. cache invalidation triggered by an update of the LDAP information).

The server gathers periodically data about its internal KPIs and exposes them through the **statistic collector interface**. These data are useful in monitoring of server functioning in production and diagnosis in case performance and service availability issues. These KPIs are:

- The rate of new requests received by the server
- The rate of requests processed with success
- The rate of requests processed with error
- The server load peak
- The rate of requests received by the server when it is in busy state
- The rate of busy responses sent to the clients
- The rate of errors when busy responses were sent to the clients
- Java Virtual Machine information (total memory, maximum memory, free memory)

One important function of an authorization service is the auditing of actions allowed or denied. The **authorization logger interface** is responsible of collecting the authorization decision taken by the server.

The server's PDP uses as input in its decision algorithm the information provided by other XACML specific entities:

- *PolicyStorage* interface is used to access the repository of access control policies administrated by PAP
- The user information enhanced with RBAC data (e.g. the Role associated to him/her) are obtained from PIP through the *RBACUser* and *Role* interfaces.

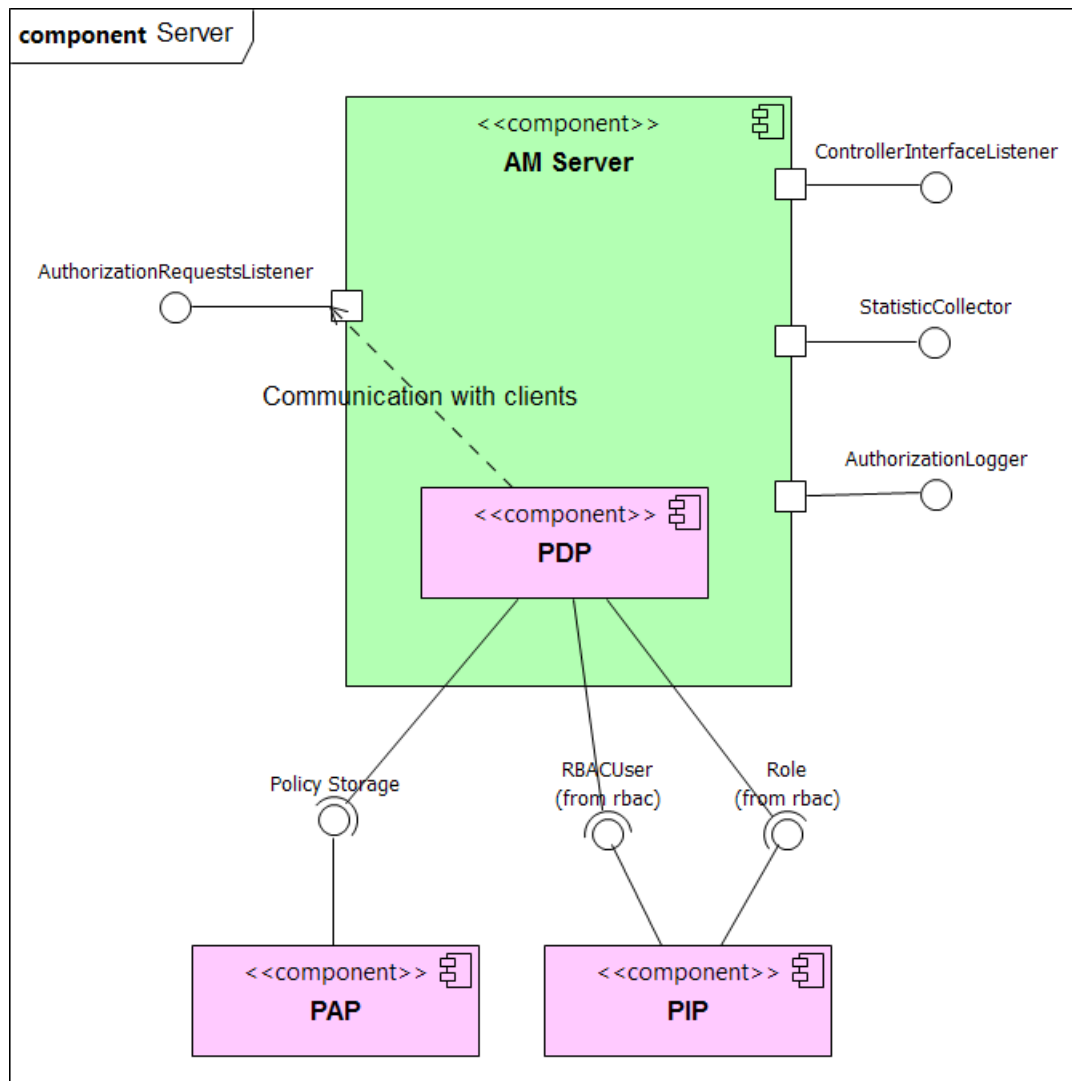


Figure 53 AM Server high level design

6.2.2.2 Client API

The client API is designed to facilitate the interaction between the clients of AM service (e.g. TDAQ components) and the AM server. The implementation of this API is offered in two flavors (Java and C++) to address the needs of the majority of client's languages.

The API is structured in 2 sets of classes and interfaces to help a client in the implementation of its access control steps:

1. authorization request preparation
2. get the authorization decision from the AM server

The preparation of an **authorization request** consists in gathering the following information necessary to take an authorization decision:

- *who request the access* authorization for the resource under protection. Sometimes the user asking the permission is not the same with the user accessing the resource, so the additional information about the user *who will access* the resource is necessary. The RequestorInfo classes described in Figure 54 and Figure 55 are in charged with the subject definition.
- *what resource* is going to be accessed and *what action* to be performed on the resource. The resources and actions definitions are very specific to the client's needs and, to ease

their integration with the AM service, special Resource and Action classes are envisaged for each TDAQ component. Figure 56 lists the resource and actions definitions for TDAQ components integrated with the AM service. Note that some resources have a limited set of actions which are not offered as individual classes, but as resource properties.

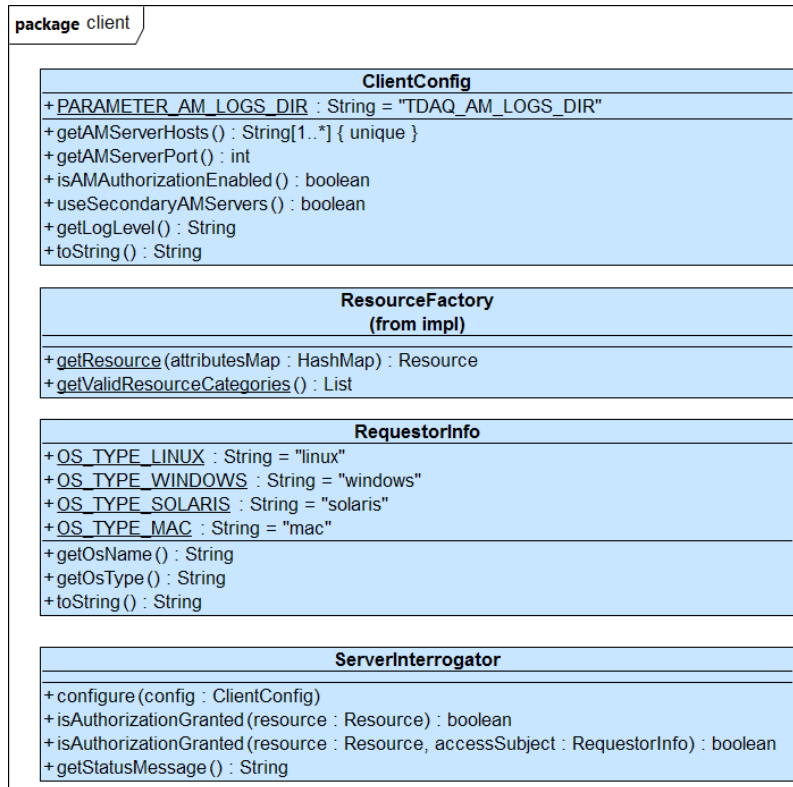


Figure 54 AM Java client API

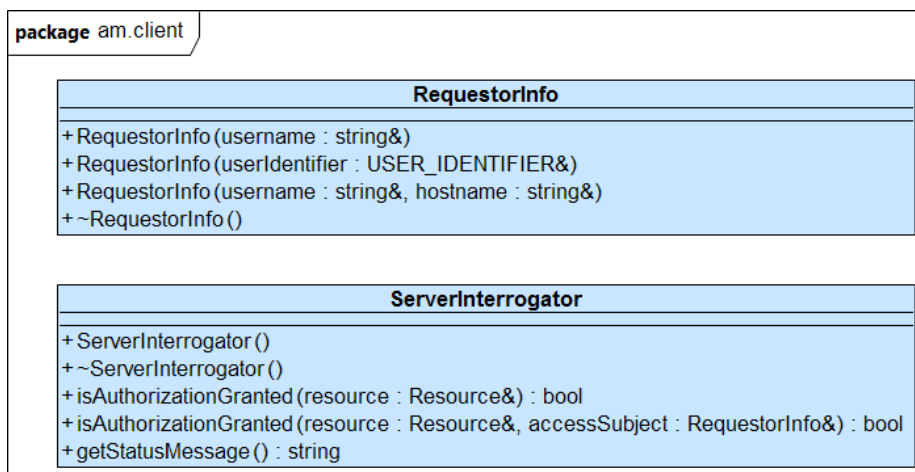


Figure 55 AM C++ client API – server interrogation

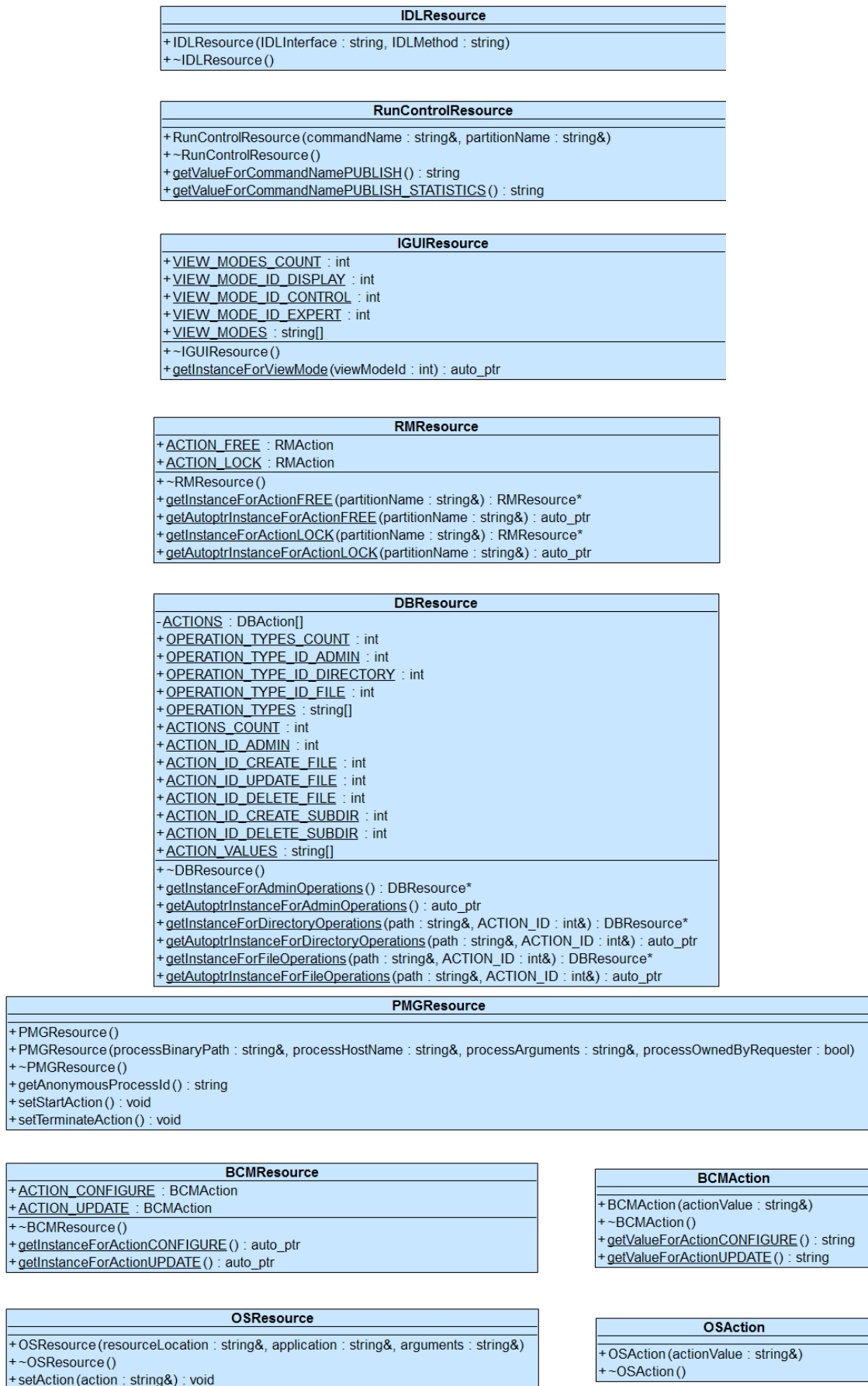


Figure 56 AM C++ client API – resource types

Once the subject, resource and actions are known for an authorization request, the **decision** is retrieved from the AM server thanks to the *ServerInterrogator* interface (Figure 54 and Figure 55). The implementation of *ServerInterrogator* interface does the job necessary to obtain a clear authorization decision:

- initiates the communication with the server
- assembles the authorization request in the format known by the server: XACML
- sends the request to the server and waits for an answer
- decodes the answer to return a *true/false* decision
- handles errors that occur in the communication with the server and tries to recover from unsuccessful conversation with the AM servers; implements fall back mechanism

The UML sequence diagram in Figure 57 is an example of client API usage to authorize the start of a TDAQ process by the TDAQ Process Manager (PMG) component. The PMG component uses the resource type special designed for its use case: PMGResource has properties meaningful for a TDAQ process and the two possible actions (start/terminate). Note that it is the PMG component which enforces the decision obtained from AM server: the process is started by PMG only if the answer from AM server is positive.

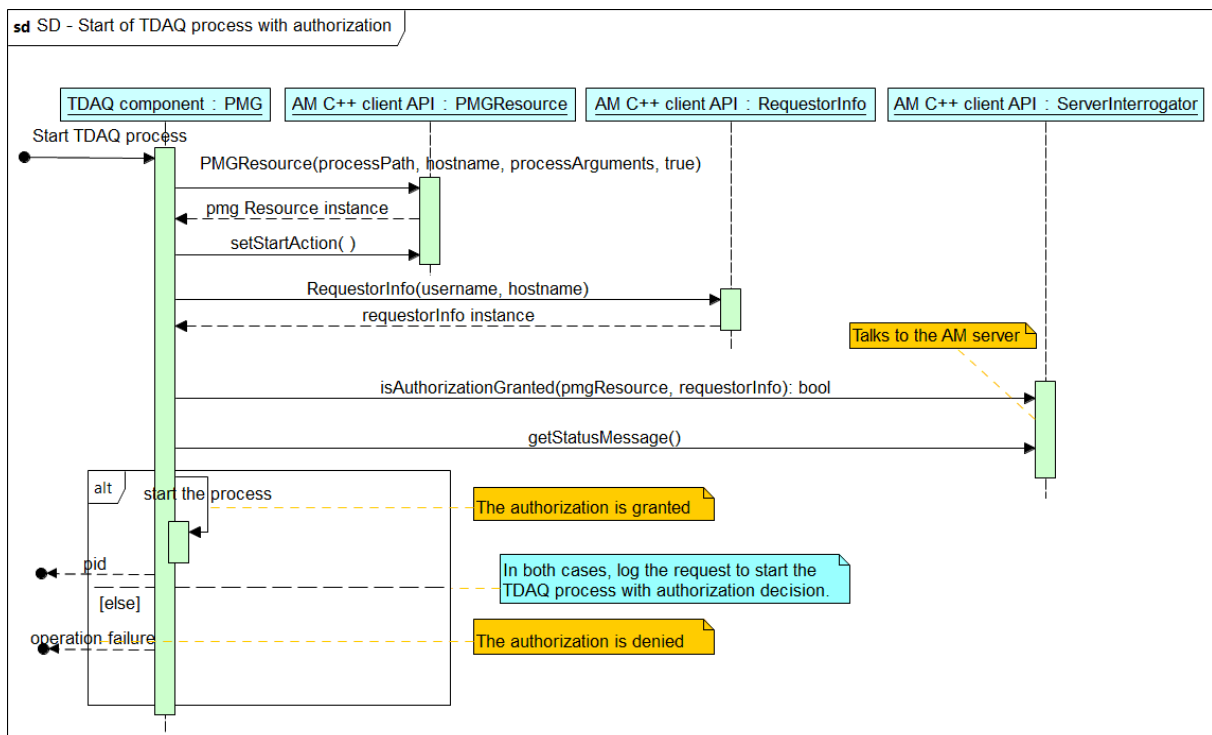


Figure 57 Example of how to use AM C++ client API

6.2.2.3 Non-functional aspects

The AM server is stateless from the point of view of communication with the client. If the client-server communication session drops unexpectedly, then the client must start over again by initiating a new communication session. This retry mechanism is the responsibility of client API implementation and must be transparent for the TDAQ component which plays the client role. The sequence diagram for retry mechanism is presented in Figure 58.

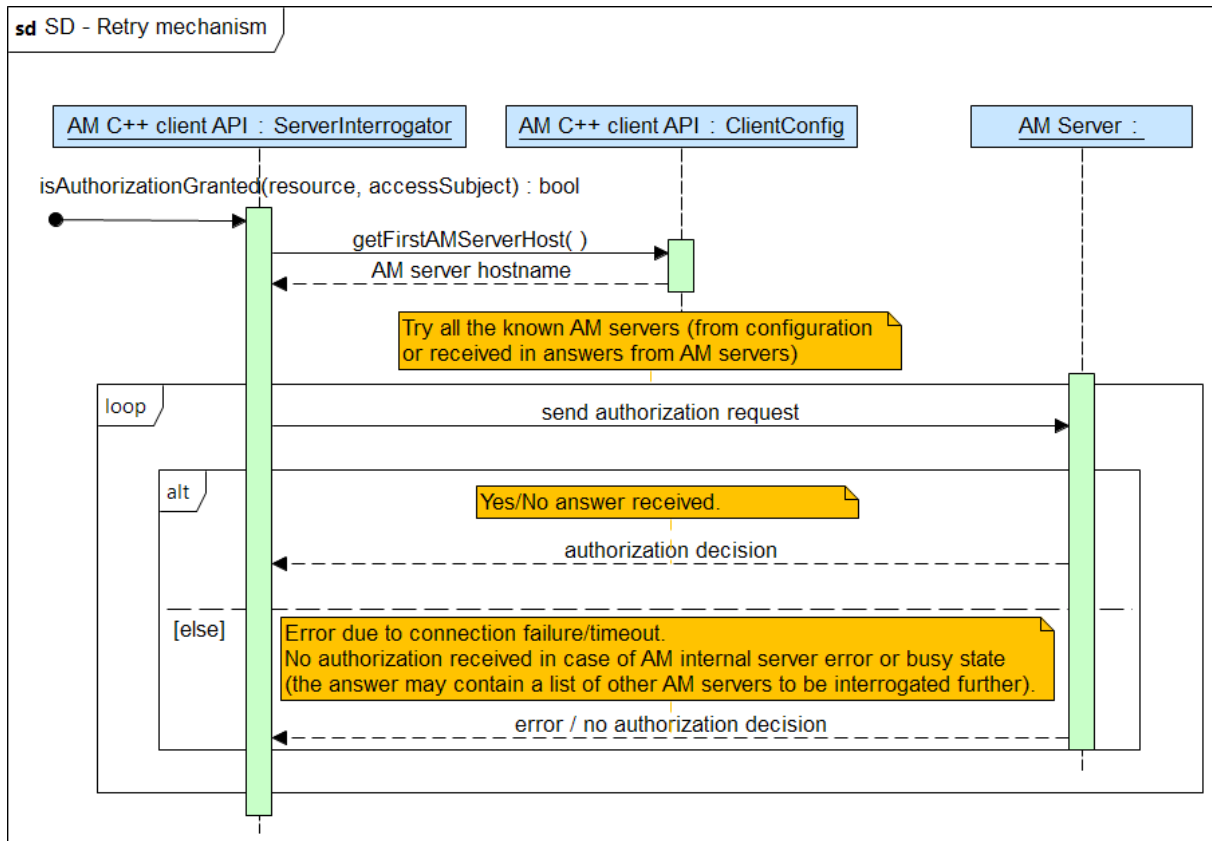


Figure 58 Retry mechanism in client API

The High Availability (HA) and load balancing characteristics of AM service are designed as a combination of retry mechanism on the client API side and a cluster of AM servers split in two levels by Domain Name Server (DNS) means. CERN IT provides a Round Robin DNS [57] service which supports the assignment of one host name to more than one Internet Protocol (IP) address. These IP addresses are returned in a round robin manner each time the host name is resolved by the DNS service.

This allows the splitting of AM servers in two levels: primary AM servers sitting behind one host name and secondary or backup AM servers defined on another host name. The client API is configured to access two AM server host names: the one corresponding to the primary set of AM servers and the one assigned for the backup AM servers.

The AM service availability and performance depends also on how fast the AM servers can process an authorization request. If a server becomes overloaded, then the response time increases and the clients may time out before receiving the authorization decision. This leads to delay on the client side and waste of processing power on the server side. To avoid such situations, an overload detection mechanism is necessary to preserve the AM service quality in terms of speed of answers to clients. The mechanism consists in monitoring the server's processing rate and, when reaches a configurable threshold, the server answers immediately to the clients request with a "busy" status until the processing rate decreases in the safe limits. Besides the "busy" state, the answer can include a list of other AM servers (e.g. the backup ones) to be interrogated by the client in another attempt to resolve its authorization request.

The mechanisms described above offer the necessary flexibility in customization of a production setup for the ALTAS Online cluster and TDAQ software needs.

6.2.3 Requirements coverage

The high level functional requirements are covered by the two main components of AM service: PAP and AM server together with client API. The coverage of use cases is outlined in the previous chapter.

The fulfillment of non-functional requirements is summarized in the table Table 15.

Table 15 Non-functional requirements coverage by TDAQ AM

Non-functional requirement	Fulfilment
Availability	Setting up the AM service as a cluster of AM servers with primary and backup layers insures the service availability in case of malfunctions of individual AM server nodes. Moreover, a client API flag to disable temporary the authorization with AM servers can be used as a safe measure if there is a major problem with AM service (e.g. network problems which make the AM cluster not accessible).
Flexibility	The AM client API allows for future integration of new clients. The command line tool used now for checking authorizations on the OS resource categories can be used in future with other OS specific shell scripts.
Portability	The AM server is developed in Java which by its nature is OS agnostic, hence easily portable to OSs where Java Virtual Machines are available. The client API is developed on both Java and C++ offering alternatives in two of the most used programming languages these days.
Performance	The responsibility is split between the AM server and the clients making the requests: <ul style="list-style-type: none"> - Server makes sure that takes the authorization decision is taken as fast as possible - Clients making the requests should not abuse of server availability and should be designed in such way to minimize the calls to the server.
Reliability	The overload detection mechanism implemented in the server guarantees that the each server performance is optimal and any extra requests beyond server limit are delegated to the other servers from the AM cluster.
Robustness	AM server is designed to handle incorrect requests from the clients or clients not being able to finish the conversation. The unavailability of user or roles information from LDAP does not stop the service functioning and, as soon as the directory service is back online, the AM service continues normally its execution. If the policies files are not available or corrupted, the AM service denies any authorization request with the appropriate reason and recovers to normal functioning as soon as the input XACML files are restored.
Scalability	The AM Server is designed to scale horizontally by increasing the processing power with the addition of new AM server instances in the AM server cluster.

6.3 Implementation

This chapter covers the implementation details of Policy Administration Point component, AM server and client API (Java and C++). We developed the applications in Java, C++ and Bash shell scripting languages suitable for SLC environment and integration with TDAQ software.

The Sun's basic implementation of XACML standard [58] is used in our AM components to encode and decode the XACML policy files, authorization requests and answers formatted in XACML, and to compute the final authorization decision. We implemented the RBAC model on top of XACML in the Policy Administration Point following the design presented in previous chapter.

6.3.1 Policy Administration Point

The PAP is developed in Java as a standalone tool which can be run thanks to a bash shell script. The internal design of PAP Java application is shown in the UML component diagram in Figure 59 where main packages are outlined:

- A set of common packages used also later by the server implementation:
 - `am.config`: Configuration interface backed by implementation to get the configuration parameters from environment variables or configuration files.
 - `am.util`: gathers packages for logging, common constants shared between server implementation and PAP, the AM specific exception class.
 - `am.rbac`: contains the interfaces specific to RBAC model entities.
 - `am.xacml`: defines an additional layer of classes specific to XACML - Hierarchical RBAC to be used on top of the basic XACML implementation (`sunxacml`)
- The `am.pap` packages contain the main classes to drive the policies assembling and generation in XACML format:
 - `am.pap.input` package provides the classes to read the permissions and permissions assignment to roles from the AM specific input format
 - `PAPtoFiles` class implements the workflow described in the sequence diagram in Figure 60.
 - `PPBuilder` class is in charge with the preparation of Permission Policies and Permission Policies Sets
 - `RPS_PPS_Builder` class manages the transformation of roles and roles hierarchies in structure specific cu XACML policy model
 - `am.pap.storage` package provides the classes to be used for XACML policies storage.
- The `am.pip` package is in charge with retrieving information about roles and roles hierarchies from LDAP. They are returned as Roles and Roles sets from `am.rbac` package.
- The `com.sun.xacml` package represents the Sun's implementation of XACML standard and is used by our implementation to generate the raw XACML data into files from our XACML RBAC model.

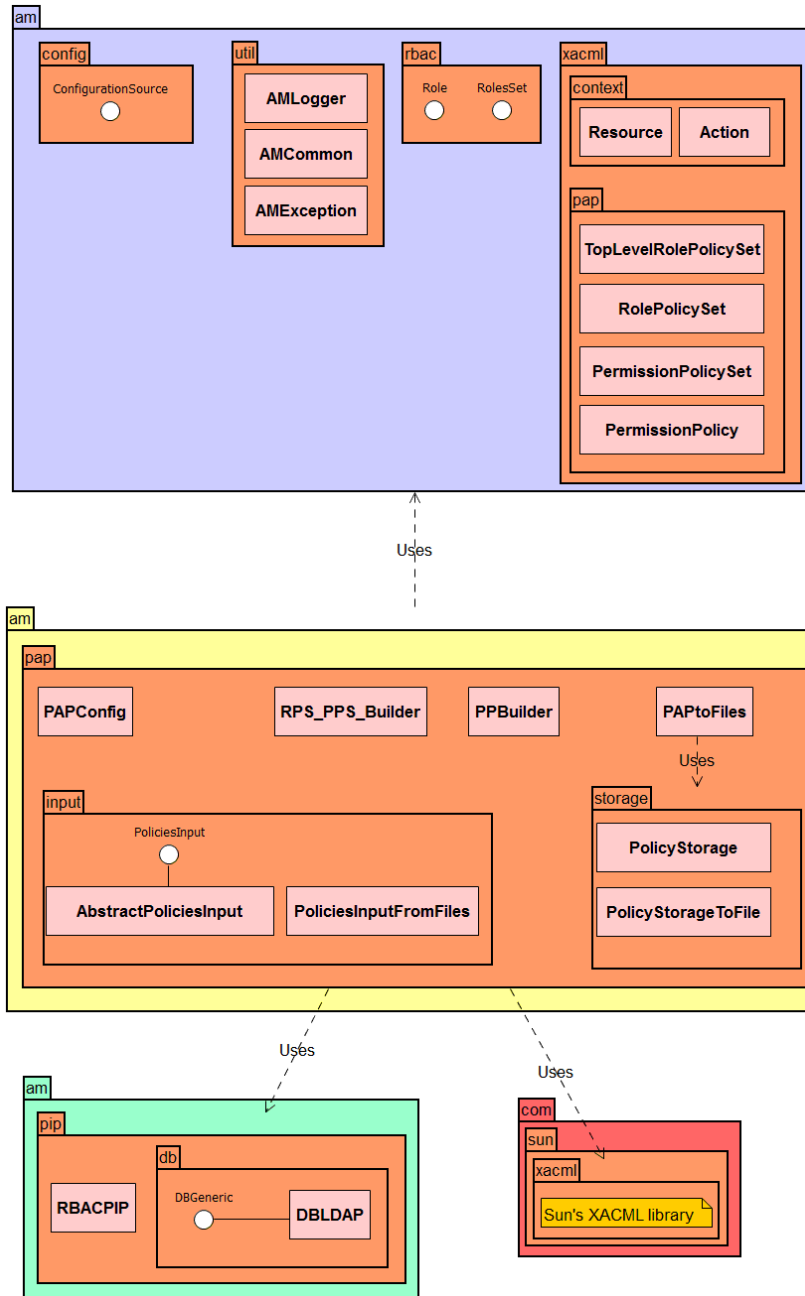


Figure 59 Access Manager Policy Administration Point – Component diagram

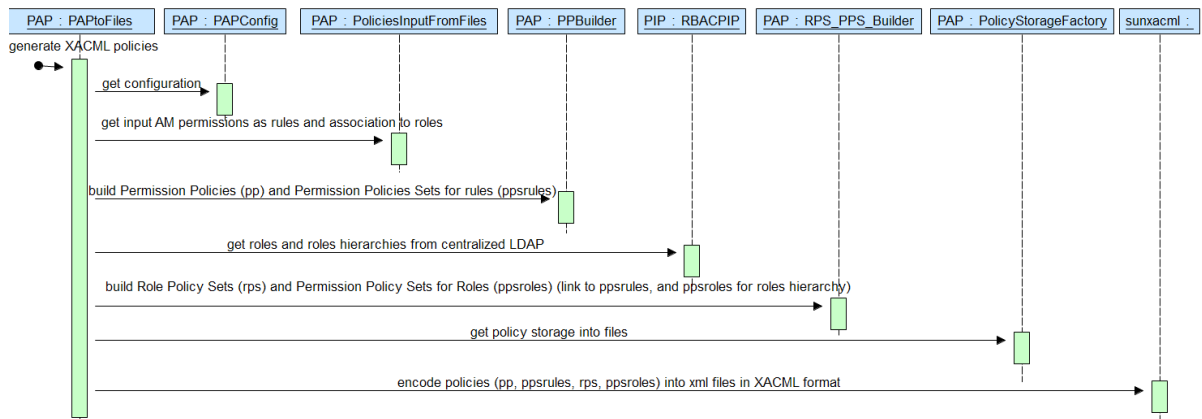


Figure 60 Sequence diagram for XACML policies generation in PAP

The PAP component can be executed by calling the helper shell script `amPAP` which has the following help screen:

```

$ ./amPAP -h
amPAP, version $Revision: 1.10 $
Run the AM's Policy Access Point to generate the policies into files.
Usage: amPAP [-d LDAP_host] [-b LDAP_basedn] [-P policies_path] [-L log_dir] [-l
log_level] [-s] [-D policies_dest_dir] [-x external_jar_classpath] [-h]
-d the LDAP server name
  Default is [localhost]
-b the LDAP base DN
  Default is [ou=atlas,o=cern,c=ch]
-P the policies directory where the XACML policy files are stored
  Default is [/home/mleahu/amsrv/data/AccessManager/policies]
-L the log directory where the log files are written
  Default is [/home/mleahu/amsrv/logs/]
-l the log level. Can be one of the following:
  NONE           - no logs at all
  MINIMUM        - log the errors and warnings
  NORMAL         - log the information messages
  VERBOSE        - log the configuration messages
  DEBUG          - log the debug messages
  ALL            - log all the messages
  Default is [NORMAL]
-s show all the AM server configurations from LDAP
-D copy the policies from policies_path to the policies_dest_dir via ssh on all
AM server nodes.
  The policies_path directory must be available on the AM server nodes!
  ONLY THIS OPERATION WILL BE PERFORMED, NO OTHER POLICY GENERATIONS!
-x external JAR classpath
-h this info
    
```

A sample execution of this script with the output generated for a given set of AM permissions and roles hierarchy is listed in the Appendix 8.7.

Once generated, the XACML policies are pushed to the AM servers running in the ATLAS Online cluster. The workflow of policies deployment on AM servers is described by the sequence diagram in Figure 61. The successful execution of this procedure relies on the LDAP centralized description of AM servers deployed in the production setup.

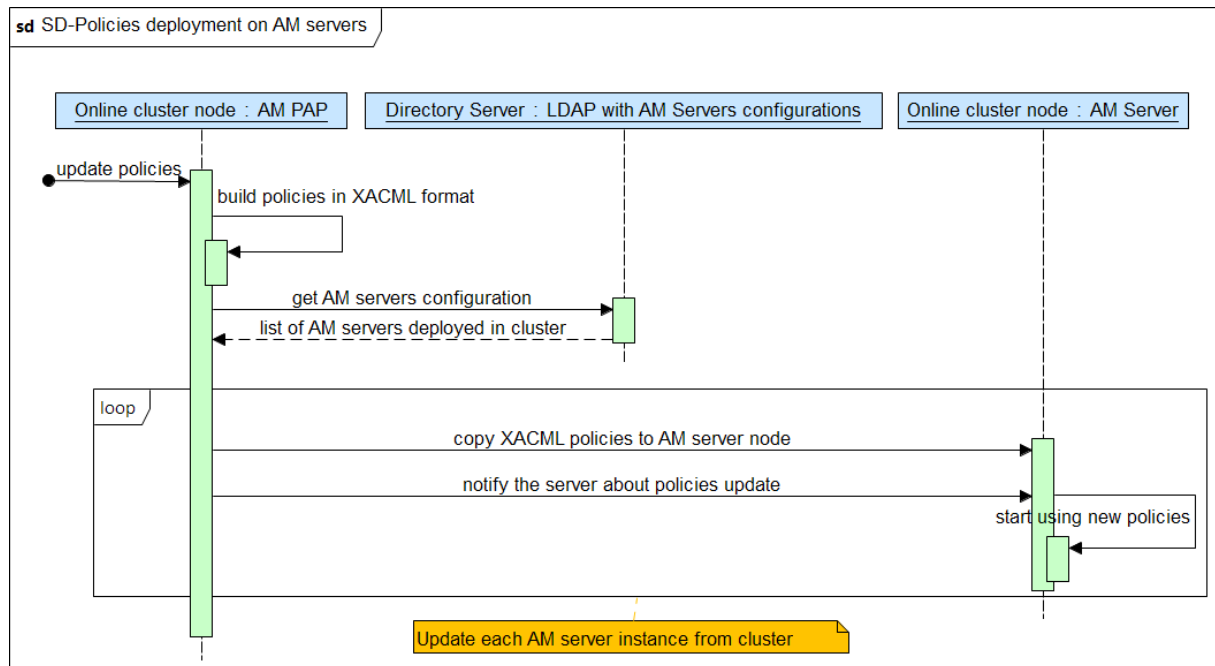


Figure 61 Build and deployment of XACML policies to AM servers

6.3.2 Server

The AM server is implemented in Java language using the Sun's XACML implementation as underlying layer for handling XACML protocol. The low level design of AM server is shown in the UML component diagram in Figure 63. The following chapters present the important implementation details of the interfaces described in the design chapter.

6.3.2.1 Authorization requests listener

The server listens for incoming network connections from clients to receive the authorization request over TCP/IP protocol, process them and sends back the answer over the same network connections. This is the key functionality of AM server and the challenge to overcome is the performance of its implementation, both from throughput and latency point of views: the server should handle as many concurrent requests from clients as possible and the time spent to process a request should be minimum.

First version of this functionality implementation was based on threads dedicated to each client connection which was handled with blocking server sockets from `java.net` package. The performance tests run with this implementation were found not satisfactory (details in chapter 6.4), although improvements have been obtained by fine tuning the threads, JVM and sockets configurations.

We reviewed the low level design of this interface implementation and the second approach led to better results as outlined in the performance test chapter 6.4. The package `am.server.NIOReactor` is the implementation of reactor pattern¹ on top of `java.nio` package. Figure 62 shows how the client's connections are handled by the reactor pattern implementation:

- The *selector* provides notifications when interests IO operations are ready for one or more handlers which subscribed in advance. The IO operations are specific to TCP network sockets (accept, read, write, close).
- The *dispatcher* receives the notifications from the selector and routes the events to the right handler. Each handler runs in a dedicated thread managed by a thread pool.
- The handlers running in threads build iteratively the request received over the network. Once the request has been assembled, it is forwarded further to a processor from the `am.pdp` package.
- The response prepared by the processor is then returned by the handler to the client over the same socket used by the *selector* in first place.

The `am.pdp` package gathers the clients request processors which have the role of PDP from XACML data flow. They are implementation of `InputHandler` interface:

- `PolicyDecisionPoint` runs the algorithm to take the authorization decision for the client request. It uses:
 - `FinderModuleForAttributeInRBACPIP` to obtain the user's roles from PIP (`am.pip` contains the link to LDAP users repository)
 - `FinderModuleForPolicyStorage` to access the XACML policy files with support from `am.pap.storage` package.
 - PDP's implementation from `com.sun.xacml` package to compute the decision based on the input from the finder modules described above.
- `PolicyDecisionPointBUSY` prepares a default "busy" answer for the clients. This is used by the overload protection mechanism detailed below.
- `PolicyDecisionPointOK` prepares a default "ok" answer used for testing purposes.

¹ According to [Wikipedia](#): *The reactor design pattern is an event handling pattern for handling service requests delivered concurrently to a service handler by one or more inputs. The service handler then demultiplexes the incoming requests and dispatches them synchronously to the associated request handlers.*

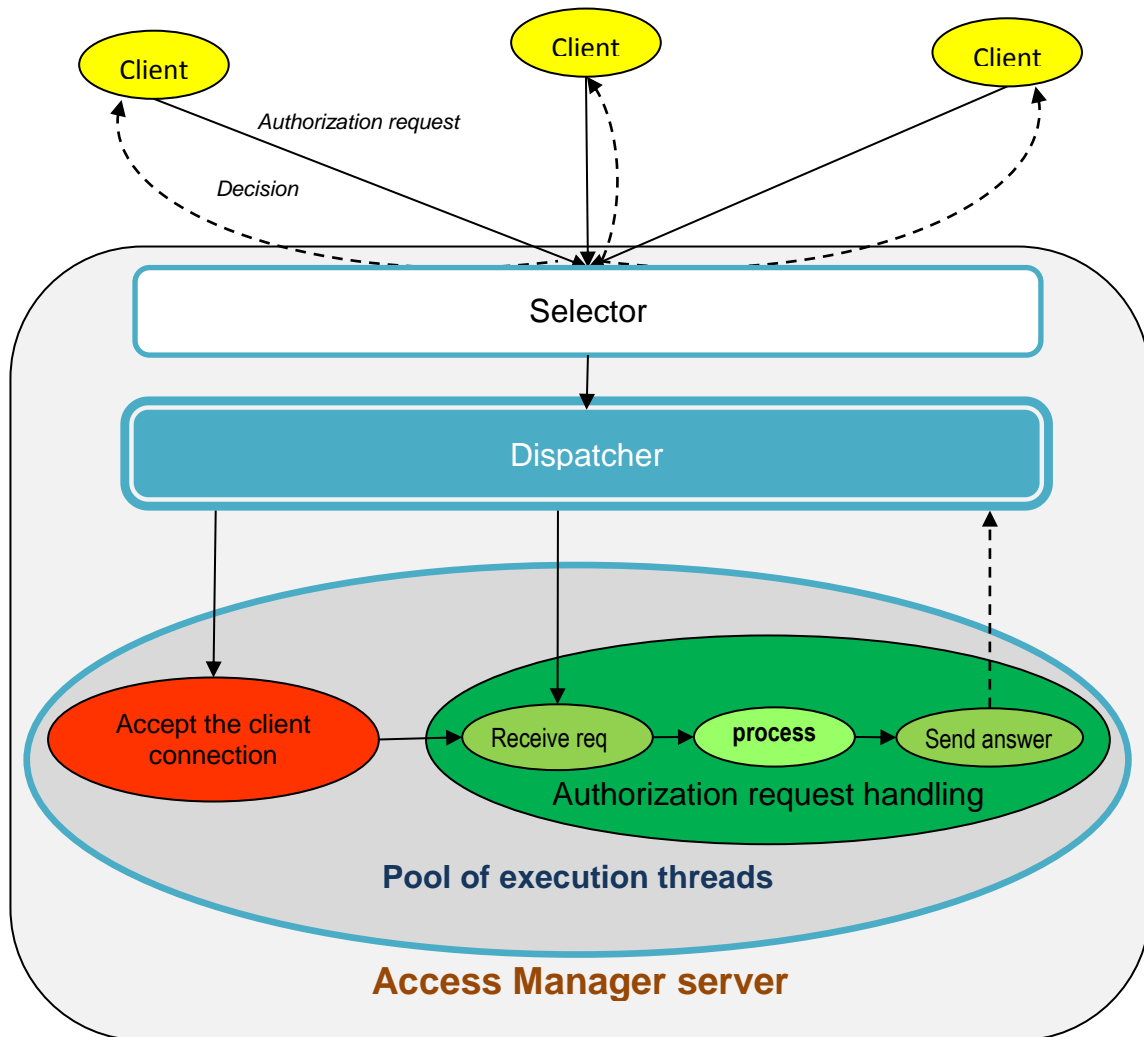


Figure 62 Reactor design pattern in AM server

The overload protection is handled by `am.server.NIOReactor.PerformanceMonitor` which monitors the server's processing rate. When the threshold is reached, the reactor's dispatcher calls `PolicyDecisionPointBUSY` to send an immediate pre-formatted "busy" message to the client including also a configurable list of alternative AM server hostnames.

We have improved more the authorization handling performance by employing caches for the PDP's inputs:

- User's information cache: there are high chances to have more than one request for the same user in short period of time, so caching user information saves many expensive LDAP interrogations. `am.pip.RBACPIP` uses an internal cache for LDAP information and `am.pdp.FinderModuleForAttributeInRBACPIP` stores the PIP evaluation result for a user also in a cache.
- Policies cache: the processing of XACML files and preparation of policies object is expensive, hence their caching in `FinderModuleForPolicyStorage` is necessary.

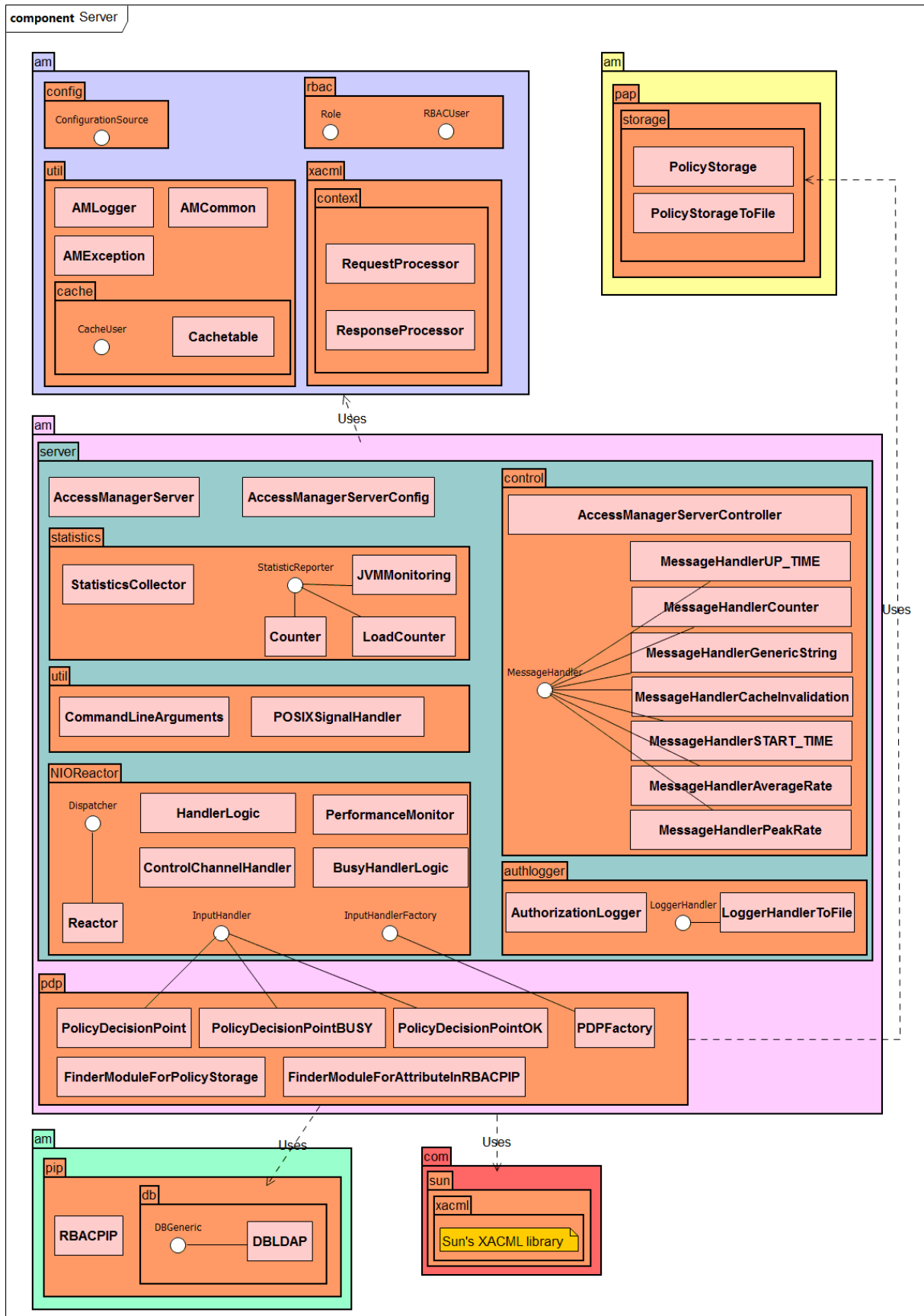


Figure 63 Access Manager server - Component diagram

6.3.2.2 Controller interface and statistics collector

The controller interface is implemented in the `am.server.controller` package. The network connections from clients who interrogate this interface are managed by the reactor pattern described in the previous chapter. `am.server.NIOReactor.ControlChannelHandler` is the reactor handler for commands to be routed to the controller interface.

`am.server.control.AccessManagerServerController` handles the commands received on the controller interface. Each implementation of `MessageHandler` interface is in charge with the processing of a command known by the server:

- `MessageHandlerCounter`, `MessageHandlerAverageRate`, `MessageHandlerPeakRate`: returns to the caller the value, average rate and peak rate of servers's KPIs:
 - Rate of new requests received by the server
 - Rate of requests processed successfully
 - Rate of requests processed with error
 - Rate of new requests received by the server when it is in busy state
 - Rate of responses with busy status
 - Rate of errors occurred when trying to send the busy state
 - Server load counter
- `MessageHandlerCacheInvalidation`: triggers the caches invalidation
- `MessageHandlerGenericString`: generic message handler for predefined string messages; e.g. it is used for the server identifier
- `MessageHandlerSTART_TIME`: returns the server start time
- `MessageHandlerUP_TIME`: returns the server up time

This interface is intended to be used by the ATLAS Online cluster monitoring system to check periodically the AM server status and trigger alarms in case of troubles.

The package `am.server.statistics` is in charge with the collection of statistics information about server's functioning and logging them periodically. Basically, the same counters exposed on controller interface are also written in log files to keep the history of server's behavior. This is especially useful during performance tests and analysis in case of server malfunctioning.

6.3.2.3 Authorization logger

The package `am.server.authlogger` implements the interface for authorization logging. The current implementation of `LoggerHandler` is the class `LoggerHandlerToFile` which writes the authorization decisions into a text file using the java logging API. The integration of authorization logging with others systems (e.g. Log Servers) can be easily done with a special implementation of `LoggerHandler` interface.

`AuthorizationLogger` is the collector of information concerning one authorization requests. It is called in various points in server classes along the authorization request handling flow to assemble the final authorization log message. When the handling of authorization request is finished, the information collected so far is finally passed to the `LoggerHandler` to be logged. The authorization log message contains the following information:

```
[date] decision client subject resource action reason
```

The following examples show two real messages logged by the authorization logger:

```
[071203 09:00:28,100] ALLOWED Client[137.138.130.220:34518] [access-subject (subject-id=mleahu) (authn-locality:dns-name=lnxatd79.cern.ch) (authn-locality:ip-address=137.138.130.220)] requests access to [(resource-category=pmg) (resource-id=)] for [(action-id=start)].

[071203 09:00:42,536] DENIED Client[137.138.130.220:34520] [intermediary-subject (subject-id=mleahu) (authn-locality:dns-name=lnxatd79.cern.ch) (authn-
```

```
locality:ip-address=137.138.130.220)access-subject (authn-locality:ip-
address=137.138.130.220) (subject-id=UNKNOWN_USER) (authn-locality:dns-
name=lnxatd79.cern.ch)] requests access to [(resource-category=pmg) (resource-id=)]
for [(action-id=start)]. Status detail(The requestor's user name was not found in
the DB!No OS valid user [UNKNOWN_USER]!)
```

6.3.2.4 Scripts

Besides the AM server application itself, we developed also a set of bash shell scripts to manage it at the OS level: start it as a service, stop it, and interrogate its status.

The AM server can be started as a Linux service thanks to the `amServerService` script:

```
$ ./amServerService -h
amServerService, version $Revision: 1.7 $
Starts the TDAQ Access Manager server on the current machine with the configuration
stored in LDAP.
Usage: amServerService [-l ldapserver] [-b basedn] [-s] [-h]
<start|stop|status|reload>
  -l the ldapserver;      default is [localhost]
  -b the basedn;          default is [ou=atlas,o=cern,c=ch]
  -s show all the AM server configurations from LDAP
  -h the help screen
```

The AM server configuration is retrieved from LDAP and the server itself is controlled by the dedicated shell script `amServer`:

```
$ ./amServer -h
amServer, version $Revision: 1.53 $
Start or stops the Access Manager server.
Usage: amServer [-D config_LDAP_host] [-B config_LDAP_basedn] [-p am_server_port] [-d
LDAP_host] [-b LDAP_basedn] [-P policies_path] [-L log_dir] [-l log_level] [-t
proc_threads] [-s] [-a|-A] [-k server_backlog] [-S saturation_factor] [-B
secondary_servers] [-u am_server_user] [-c N_minutes] [-i jvm_heap_ini] [-j
jvm_heap_max] [-m] [-x external_jar_classpath] [-h]
<start|stop|force_stop|status|restart|force_restart>
  -D the LDAP server host name for the AM server configuration
    Default configuration source is the environment.
  -B the LDAP base DN for the AM server configuration
    Default is [ou=atlas,o=cern,c=ch]
  -p the port number on which the AM server will listen for requests
    Default is [20000]
  -d the LDAP server name
    Default is [localhost]
  -b the LDAP base DN
    Default is [ou=atlas,o=cern,c=ch]
  -P the policies directory where the XACML policy files are stored
    Default is [./../data/AccessManager/policies/]
  -L the log directory where the log files are written
    Default is [/home/mleahu/amsrv/logs/]
  -l the log level. Can be one of the following:
    NONE          - no logs at all
    MINIMUM       - log the errors and warnings
    NORMAL        - log the information messages
    VERBOSE       - log the configuration messages
    DEBUG        - log the debug messages
    ALL          - log all the messages
    Default is [NORMAL]
  -t the maximum number of AM Processing Threads
  -s enable the statistics feature
    Default is [off]
  -a turn off the authorization logger
  -A turn on the authorization logger for all the messages
    Currently is [minimum]
  -k server socket backlog size (default 200)
  -S set the saturation factor above which the server responds with SERVER_BUSY
  -B specify the secondary AM servers to be sent to the clients when responds with
SERVER_BUSY
```

```
-u the user as who the AM server will run
   Default is [mleahu]
-c append to the crontab the command to archive the server logs each N minutes
-i the initial JVM heap size. 0 to use the JVM default
   Script default is [512m]
-j the maximum JVM heap size. 0 to use the JVM default
   Script default is [512m]
-m start the JVM in debug mode for socket connection on port 8000
-x external JAR classpath
-h this info
```

Once the server is started, its controller interface can be accessed through a dedicated client shell script (and a binary behind it) `amServerRemoteController`:

```
$ amServerRemoteController -h
amServerRemoteController, version $Revision: 59448 $
Send messages to AM Servers using their LDAP configuration.
Usage: amServerRemoteController [-l ldapserver] [-H ldapuri] [-b basedn] [-s] [-t
am_server_name] <-c command> [-h]
-l the ldapserver;      default is []
-H the ldapuri;        default is [ldap://xldap.cern.ch/]
-b the basedn;         default is [dc=cern,dc=ch]
-s show all the AM server configurations from LDAP
-c the command to send to the server. The valid commands are:
  SERVER_ID           Ask the Access Manager server identifier
  START_TIME          Ask the Access Manager server start time
  UP_TIME              Ask the Access Manager server up time
  SERVER_STOP         Ask the Access Manager to shut down
  LDAP_UPDATED        Inform the Access Manager server that the LDAP
information was updated
  PAP_UPDATED         Inform the Access Manager server that the Policy
Administration Point was updated
  SECONDARY_SERVERS   Ask the list of secondary servers
  BUSY_RESP            Ask the number of busy responses
  BUSY_RESP_ERR       Ask the number of busy response errors
  NEW_REQ             Ask the number of new requests for authorization
received from clients
  REQ_PROC_ERR        Ask the number of requests processed with error
  REQ_PROC_OK         Ask the number of requests processed successfully
  REQ_WHEN_BUSY       Ask the number of new requests for authorization
received from clients when the server was busy
  AVG_RATE_BUSY_RESP  Ask the average rate of busy responses
  AVG_RATE_BUSY_RESP_ERR Ask the average rate of busy responses errors
  AVG_RATE_NEW_REQ    Ask the average rate of new requests for authorization
received from clients
  AVG_RATE_REQ_PROC_ERR Ask the average rate of requests processed with error
  AVG_RATE_REQ_PROC_OK Ask the average rate of requests processed successfully
  AVG_RATE_REQ_WHEN_BUSY Ask the average rate of new requests for authorization
received from clients when the server was busy
  PEAK_RATE_BUSY_RESP Ask the peak rate of busy responses
  PEAK_RATE_BUSY_RESP_ERR Ask the peak rate of busy responses errors
  PEAK_RATE_NEW_REQ   Ask the peak rate of new requests for authorization
received from clients
  PEAK_RATE_REQ_PROC_ERR Ask the peak rate of requests processed with error
  PEAK_RATE_REQ_PROC_OK Ask the peak rate of requests processed successfully
  PEAK_RATE_REQ_WHEN_BUSY Ask the peak rate of new requests for authorization
received from clients when the server was busy
-t send the command only to this AM server. Default: send the command to all AM
servers configured in LDAP.
-h the help screen
```

This script uses also the AM server configurations defined in LDAP to reach them on the right hostname and port number.

The script `amServerLDAPNotifier` is necessary to run as a service on each LDAP server with users and roles information deployed in the ATLAS Online cluster. The script responsibility is to notify

all AM servers each time the LDAP information is updated so that the AM servers invalidate their user's cache.

```
$ amServerLDAPNotifier -h
This shell script takes care of starting and stopping the dnotify daemons to notify
the AM servers about the LDAP update events
Usage: amServerLDAPNotifier {start|stop|restart|status}
```

6.3.3 Client API

The client API is implemented in two flavors (a Java jar library and a C++ library) because the TDAQ components are implemented in both Java and C++. However, the format of messages exchanged between the client and the server is the same in both implementations.

The authorization request is prepared by the client API in the XACML format including the subject information, the resource to be accessed and the action to be performed on the resource. A sample authorization request for user `mleahu` on the resource `os/gateway/login` with action `remote` is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
  <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-
address" DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>127.0.0.1</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-
locality:dns-name" DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>localhost.localdomain</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>mleahu</AttributeValue>
    </Attribute>
  </Subject>
  <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-
category:intermediary-subject">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-
address" DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>127.0.0.1</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-
locality:dns-name" DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>localhost.localdomain</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>mleahu</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
type:application" DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>login</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>gateway</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
category" DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>os</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
```

```
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
  <AttributeValue>remote</AttributeValue>
</Attribute>
</Action>
<Environment />
</Request>
```

The response from the server is also in the XAML format. The client API implementation decodes the response into a true/false or error status for the API caller. A sample answer for the authorization request above is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Result ResourceId="gateway">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
    </Status>
  </Result>
</Response>
```

The component diagram of Java client API (Figure 64) outlines the following main packages:

- `am.xacml.context`: contains the implementation of entities which compose an authorization request; the resources and actions are specialized for the TDAQ components which are integrated with AM service. The helpers for assembling a request (`RequestProcessor`) and decoding a response (`ResponseProcessor`) use the basic XACML implementation from Sun's XACML package (`com.sun.xacml`). This package is used also by the Policy Administration Point in its task of policies preparation and storage into files.
- `am.client`: offers the classes for preparation of requestor information and the authorization request; it contains also the algorithm for communication with the server over network and the retry mechanism (see Figure 58)

The C++ client API follows the same structure (see component diagram in Figure 65) as Java client API, but with a difference: the classes in `am.xacml` package (except `am.xacml.impl` which are AM specific) implement the low level details of XACML standard for the request encoding and response decoding; they correspond to the `com.sun.xacml` package in the Java client API.

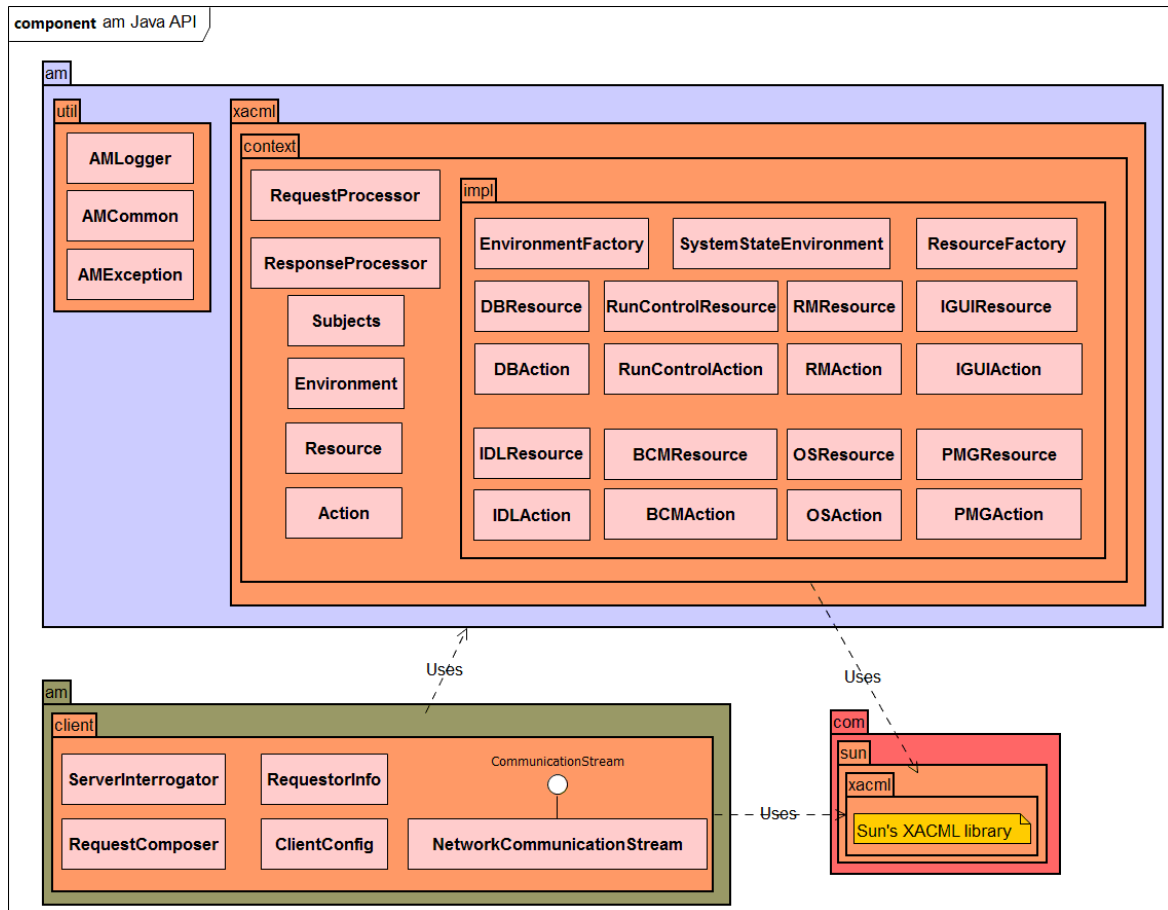


Figure 64 Component diagram of AM Java client API

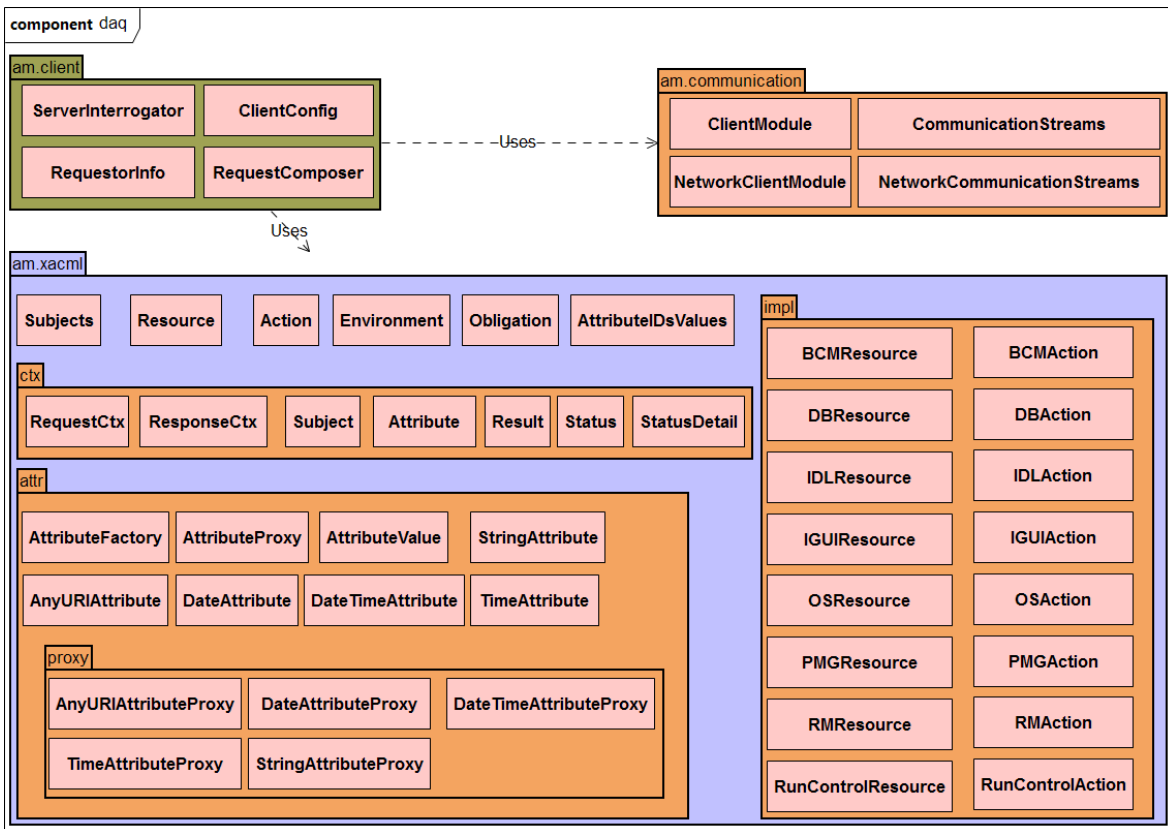


Figure 65 Component diagram of AM C++ client API

6.4 Tests

The TDAQ AM service has undergone a series of tests focused on functionality and performance before its deployment in the ATLAS Online cluster together with the TDAQ software. The following chapters detail the testing strategies and the results obtained.

6.4.1 Functional Tests

The functional tests have been run in two modes: in isolation and integrated in the TDAQ software. Their goal was to validate the correct functioning of AM service for the use cases described in the requirements chapter 6.1.

The functional test executed in **isolation** used the following test setup:

- one PAP instance which prepared the XACML policies based on input file (`AMRules.txt`) listed in appendix 8.7.1
- one AM server configured with the XACML policies generated in the previous step and listed in appendix 8.7
- one LDAP server (OpenLDAP) ready to accommodate RBAC data (configuration done according to instructions from 4.2.1):
 - roles hierarchy defined in LDAP is the same as in appendix 8.7
 - user defined in LDAP to be used as the subject asking for authorization in the tests
- one Java application developed on top of JUnit 4 framework [59] which uses the Java client API to issue authorization requests to the server

The tests run with this setup consisted in preparing automatically a set of authorization requests corresponding to the rules defined in the `AMRules.txt` and asking the server to process them. The results were compared with the expected outcome according to the roles enabled for the user used in the tests.

The **integration** of AM service in the TDAQ software implied also the inclusion of TDAQ AM service in the large scale tests executed on TDAQ software releases in the test lab cluster. This test phase validated that integration process was successful and that the service provided the expected functionality.

6.4.2 Performance and stress tests

The **performance tests** were run to assess the AM service processing speed and find out the best configuration parameters to be used in the production installation.

The hardware configurations of the nodes from the test lab used in the performance tests are the following:

- AM server nodes:
 - Processor: Intel(R) Xeon(TM) CPU 3.06GHz
 - RAM : 2 GB
 - Operating System: Scientific Linux Cern 4 – x86
 - Java Runtime Environment 5
- Client nodes:
 - Processor: Intel(R) XEON(TM) CPU 2.00GHz
 - RAM: 512 MB
 - Operating System: Scientific Linux Cern 4 – x86
 - Java Runtime Environment 5

In order to simulate a high number of authorization requests sent to the AM server, we developed a set of tools composed of:

- C++ application which uses the C++ client API to issue authorization requests to the target AM server. This test client application allows to:
 - Repeatedly send an authorization request to the server; number of iterations is configurable
 - Insert delays between consecutive authorization requests
 - Count the time spent to send all the requests and computes the average

The helper bash script `amServerInterrogatorCTest` calls the test client application with following options:

```
amServerInterrogatorCTest, version $Revision: 1.13 $
Run the AM's Server Interrogator implementation using the C++ API
'AccessManagerServerInterrogatorTest':
Access Manager Server Interrogator test application for C++ API.
Version: $Revision: 53652 $ built on Jul 16 2012 22:22:36
Client test implementation version: $Revision: 85994 $

Usage: amServerInterrogatorCTest [-a <on|off>] [-s server_host] [-p server_port] [-l
log_level] [-n number_iterations] [-t delay_iterations] [-S] [-r resource_type]
[-1|2|3|4 resource_attr] [-A action] [-u access_subject_username] [-o
access_subject_hostname] [-h]
  -a enable or disabled the Access Manager authorization in C++ API. Valid values
are 'on' and 'off'
  Default is [on]
  -s the hostname where Access Manager server is running
  Default is [lnxatdmon]
  -p the port number on which the Access Manager server is listening
  Default is [20000]
  -l the ERS debug level. Should be a positive number
  Default is [0]
  -n specify the number of test iterations
  Default is [1]
  -t delay between the test iterations in ms
  Default is [0]
  -S use secondary AM servers if provided
  Default is [false]
  -r the resource type to be accessed.
  -1 the resource type attribute #1.
  -2 the resource type attribute #2.
  -3 the resource type attribute #3.
  -4 the resource type attribute #4.
  -A the action name
  -u the access subject's username
  Default is []
  -o the access subject's hostname
  Default is []
  -h this info
```

- Test suite configuration which is defined as a list of test client application to be run in parallel on node. The following listing shows a sample test suite configuration:

```
# AM test suite configuration file that will be included by the amTestSuite script.
# Version
# 16/02/2007 [mleahu]: Initial version

# the test suite description to be used in the log files
ts_description="PMG test - 8 am clients per node, 501 requests per am client:"

# the estimated execution time in minutes
ts_estimated_execution_time=6

# the test suite commands to execute in parallel on each node
ts_commands=(
```

```

"amServerInterrogatorCTest -u mleahu -l 0 -s ${ts_am_server_host} -p
${ts_am_server_port} -n 501 -t ${ts_time_interval} -r pmg -l start"
"amServerInterrogatorCTest -u mleahu -l 0 -s ${ts_am_server_host} -p
${ts_am_server_port} -n 501 -t ${ts_time_interval} -r pmg -l terminate"
"amServerInterrogatorCTest -u mleahu -l 0 -s ${ts_am_server_host} -p
${ts_am_server_port} -n 501 -t ${ts_time_interval} -r rc -l publish"
"amServerInterrogatorCTest -u mleahu -l 0 -s ${ts_am_server_host} -p
${ts_am_server_port} -n 501 -t ${ts_time_interval} -r rc -l publish_statistics"
"amServerInterrogatorCTest -u mleahu -l 0 -s ${ts_am_server_host} -p
${ts_am_server_port} -n 501 -t ${ts_time_interval} -r rc -l CMD -2 PART"
"amServerInterrogatorCTest -u mleahu -l 0 -s ${ts_am_server_host} -p
${ts_am_server_port} -n 501 -t ${ts_time_interval} -r igui -l display"
"amServerInterrogatorCTest -u mleahu -l 0 -s ${ts_am_server_host} -p
${ts_am_server_port} -n 501 -t ${ts_time_interval} -r igui -l control"
"amServerInterrogatorCTest -u mleahu -l 0 -s ${ts_am_server_host} -p
${ts_am_server_port} -n 501 -t ${ts_time_interval} -r igui -l expert"
)

```

The keywords `ts_am_server_host`, `ts_am_server_port`, `ts_time_interval` are replaced at test suite execution time by the configured values.

- Shell script `amTestSuite` to schedule the execution of test suites on a set of nodes from the test lab cluster:

```

$ ./amTestSuite -h
amTestSuite, version 0.4
Runs the AM test suite
Usage: amTestSuite [-i identification_string] [-s server_host] [-p server_port] [-t
time_interval] [-l nodes_file] [-n max_nodes] [-d start_delay] [-c ts_cfg_dir] [-r
daq_relas] [-L] [-X] [-h]
  -i the identification of this test run
  -s the hostname where Access Manager server is running
     Default is [lnxatdmon]
  -p the port number on which the Access Manager server is listening
     Default is [20000]
  -t time intervals in milliseconds between authorization requests sent to server by
a client process
     Default is [0]
  -l the file with list of machines in the directory /home/mleahu/amtests/nodes-
lists on which the tests will run
     Default is [point1-cluster.lst]
  -n the maximum number of nodes from the nodes_file to use for tests
     Default is [100]
  -d the number of minutes after which the tests are scheduled to run with crontab
     Default is [5]
  -c the directory where the test suite configuration files are stored ( extension
'tscfg')
     Default is [/home/mleahu/amtests]
  -r the TDAQ release directory to use
     Default is []
  -L list the crontabs on all the nodes in the nodes_file
     Default is [0]
  -X clean the crontabs on all the nodes in the nodes_file
     Default is [0]
  -h this info

```

The `identification_string` represents the name of the file with test suite configuration stored in `ts_cfg_dir` and the list of nodes where to schedule the execution is defined in `nodes_file` (one hostname per line).

- Reports aggregation tool which collects the test reports from the cluster nodes to consolidate them for later analysis:

```

$ ./amTestSuiteReports -h
amTestSuiteReports, version 0.2
Process the AM test suite reports
Usage: amTestSuiteReports [-c ts_cfg_dir] [-x] [-h]
  -c the directory where the test suite configuration files are stored (extension
'tscfg')

```

```

Default is [/home/mleahu/amtests]
-x clean all the summary files in the reports directory reports/*.sum
-h this info

```

The test strategy consists in running in parallel on multiple cluster nodes a set of authorization requests to increase gradually the load on the AM server. The user used in test had all roles assigned and enabled to maximize the processing effort on the server. The policies were the same as in the functional tests configuration.

First set of performance tests were run on the initial implementation of AM server, the one based on threads dedicated to each client connection which was handled with blocking server sockets from `java.net` package. There were two runs:

- first run had the goal of assessing the server performance and identify points to be improved. At the same time, we ran profiling tests on the server to pinpoint the performance issues during the workflow of authorization request processing.
- second one with server improvements. The improvements consisted mainly in:
 - decrease the IO operations (e.g. reduced logging or changed log levels for some messages)
 - reduce synchronization between threads (use of `ThreadLocal` variables)
 - use of `StringBuilder` for faster strings assembling
 - loops reorganization
 - set `TCP_NODELAY` option on the sockets (both on server and client sides)
 - adjustments of server configuration: use 2 threads instead of 5, increased the initial JVM heap size from 128MB to 256MB and maximum size to 320MB

The server and client configurations used in second run are listed in Table 16, respectively Table 17.

Table 16 AM server configuration on the first set of performance tests (2nd run)

Server configuration parameter	Value
Processing threads	2
Request queue size	100
Timeout - LDAP socket connection	3 seconds
Timeout - LDAP operation time limit	3 seconds
Timeout - Message receive from the client	6 seconds
Cache timeout - PIP Evaluation Context	5 seconds
Cache timeout - RBAC users	60 seconds
Cache timeout – Policies	300 seconds
JVM initial heap size	256 MB
JVM max heap size	320 MB

Table 17 Client configuration on the first set of performance tests (2nd run)

Client configuration parameter	Value
Timeout - Socket connection to the server	2 seconds
Timeout - Message receive from the server	6 seconds

The test results from both runs are show in Table 18. The improvements implemented between two runs are significant and reflected by the following metrics:

- top processing rate increased with 76.44%: from 208 to 367
- there are no failed interrogations reported by the client at top server processing rate on the second run
- only 0.2% of clients interrogations took more than one second on the second run compared with 1.3 % from first run; both results obtained at the server top throughput
- the second run can handle more parallel clients at server top speed (96 parallel clients) compared with the first run (56)

It is worth to notice that in each run the server performs worst once its maximum throughput is reached: increasing the load makes the server inefficient and its processing rate drops while the clients experience delays or even failed interrogations due to timeouts.

Table 18 First performance test results (comparison between runs)

8 parallel client interrogations from #nodes	Server average requests processing rate/s		Interrogation duration > 1s (%)		Failed interrogations (%)	
	1 st run	2 nd run	1 st run	2 nd run	1 st run	2 nd run
4 nodes (32 p.c.)	195	NT	0.6	NT	0.3	NT
5 nodes (40p.c.)	204	NT	1	NT	0.2	NT
6 nodes (48 p.c.)	205	NT	0.8	NT	<0.1	NT
7 nodes (56 p.c.)	<u>208</u>	364	1.3	0	<0.1	0
8 nodes (64 p.c.)	201	341	3.4	0.2	0.1	< 0.1
9 nodes (72 p.c.)	170	360	6.2	0.2	1.3	0
10 nodes (80 p.c.)	152	364	12.1	0	4.1	0
11 nodes (88 p.c.)	NT	364	NT	0	NT	0
12 nodes (96 p.c.)	NT	<u>367</u>	NT	0.2	NT	0
13 nodes(104 p.c.)	NT	360	NT	0	NT	3.8

The histogram of server interrogation duration experienced by the clients in the 2nd run is drawn in Figure 66.

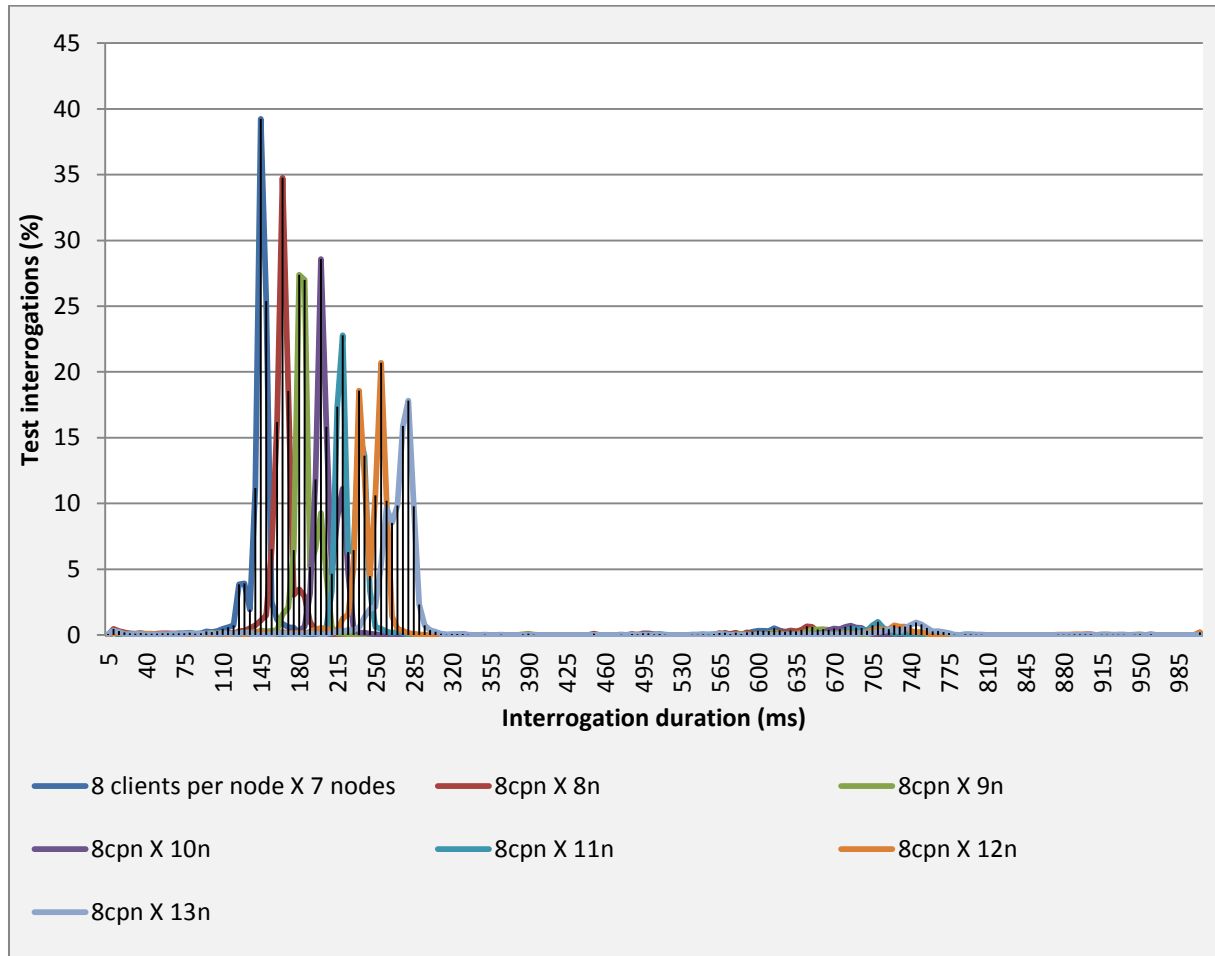


Figure 66 Histogram of first performance tests (2nd run)

Despite the major improvements reported by the second run, the AM server performance was found not satisfactory for the production environment where a more complex distribution of requests types is expected with high number of users and possible more complex policies.

Table 19 AM server configuration on the second set of performance test

Server configuration parameter	Value
Processing threads	2
Timeout - LDAP socket connection	3 seconds
Timeout - LDAP operation time limit	3 seconds
Timeout - Message receive from the client	6 seconds
Cache timeout - PIP Evaluation Context	5 seconds
Cache timeout - RBAC users	60 seconds
Cache timeout – Policies	300 seconds
JVM initial heap size	384 MB
JVM max heap size	512 MB

We started a more thorough analysis of server’s low level design and the solution based on reactor pattern with `java.nio` usage was found suitable for our needs. The details of this implementation are provided in chapter 6.3.2.

The second set of performance tests were run with the server configuration listed in Table 19. The test results listed in Table 20 show an impressive improvement:

- server top throughput (514) is 147% better than very first test (208) and 40% better compared with the previous version (367)
- zero failed interrogations at top speed
- 112 parallel clients handled by the server at top speed compared with 96 in the previous version

Table 20 Second performance test results

# parallel client interrogations	Server average requests processing rate/s	Interrogation duration > 1s (%)	Failed interrogations (%)
80	453	0.5	0
112	514	1	0
144	485	3.5	< 0.01
176	488	4.5	0.02
208	460	5.3	0.02
240	442	6.12	0.12
264	439	6.73	0.21
288	432	6.9	0.52

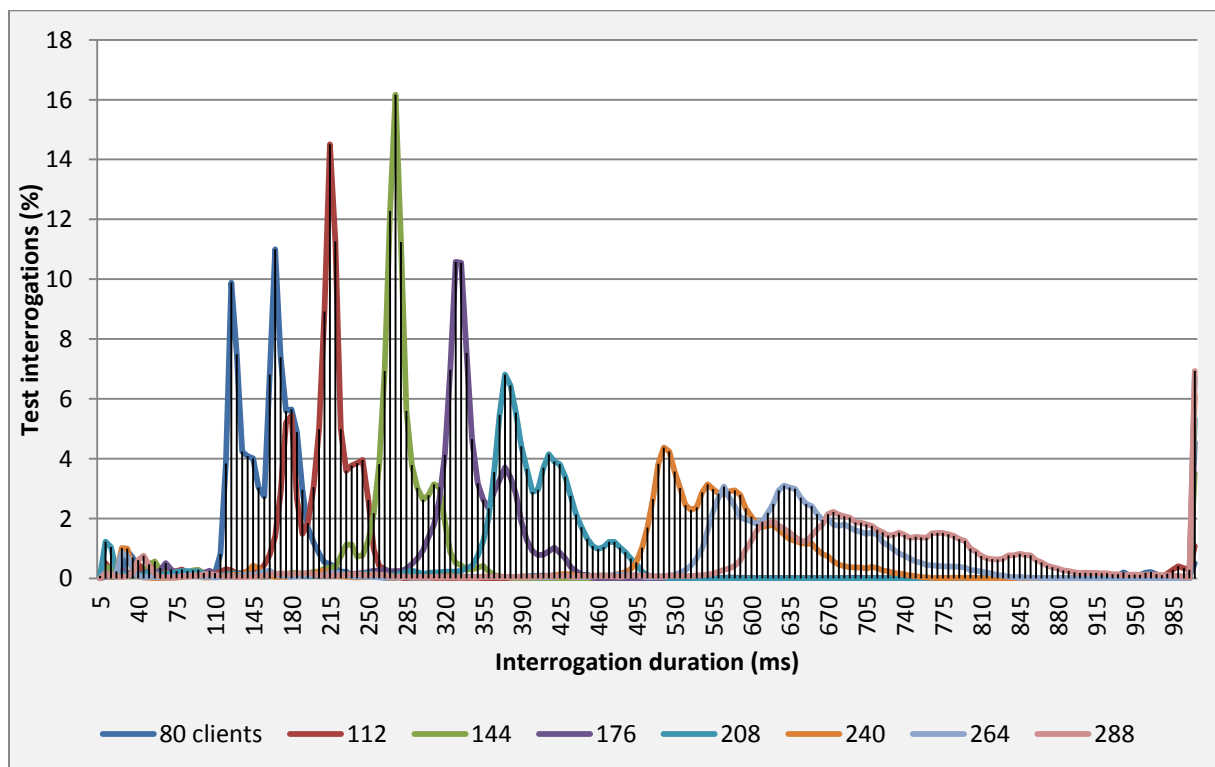


Figure 67 Histogram of second performance tests

The histogram of interrogation duration for the second set of performance tests is shown in Figure 67.

After this last round of performance tests, the following observations can be made:

- In case of 80 parallel clients:
 - Server top throughput: 453 requests /second
 - 91% interrogations took under 200 ms
- In case of 112 parallel clients:
 - Server top throughput: 514 requests /second
 - 95% interrogations took under 250 ms
- Above this limit, the server becomes overloaded and its performance decreases with increased latency on client side

We conclude that the server's optimal usage is at about 100 parallel clients with its top throughput between 450 and 500. This threshold can be set in the server configuration for the overload protection so that the server can run in optimal conditions. If there are foreseen more clients to run in parallel, then another server instance can be deployed in the AM server cluster.

The **stress tests** consisted in preparing a cluster of 2 AM servers and fully load them with requests generated by the test tools running on multiple nodes on the test lab cluster for more than 24 hours. We monitored the AM server cluster behavior with the help of server's statistics collectors. We verified the load balancing between the nodes, the behavior in case of a server simulated crash in the AM servers cluster and that the server's throughput was constant during stress test period. The tests completed successfully, the minor issues found being fixed before the installation into ATLAS Online cluster environment.

6.5 Production Setup

The deployment diagram for the AM service in the ATLAS Online cluster is depicted in Figure 68. The AM servers are organized in two clusters:

- Primary AM servers cluster sits behind the hostname `pc-tdq-onl-ams` configured in CERNT IT DNS service with round robin policy. This makes the addition of a new node to the cluster transparent for the AM clients.
- Secondary or backup AM servers cluster is defined on the hostname `pc-tdq-onl-ams-bak` configured also with round robin policy in DNS.

The AM clients (TDAQ Components integrated with C++ or Java client API, the Application Gateway and Control Room Desktop using the AM command line tool) are configured to access following AM servers: `pc-tdq-onl-ams`, `pc-tdq-onl-ams-bak`. This means authorization requests failed with any of the servers from the primary cluster are retried with another server from the backup cluster.

The following chapters describe the details of an AM server node installation with configuration in LDAP and the integration of the AM servers with monitoring system and LDAP servers for change notifications. The environment variables necessary for TDAQ components integrated with AM service are listed at the end.

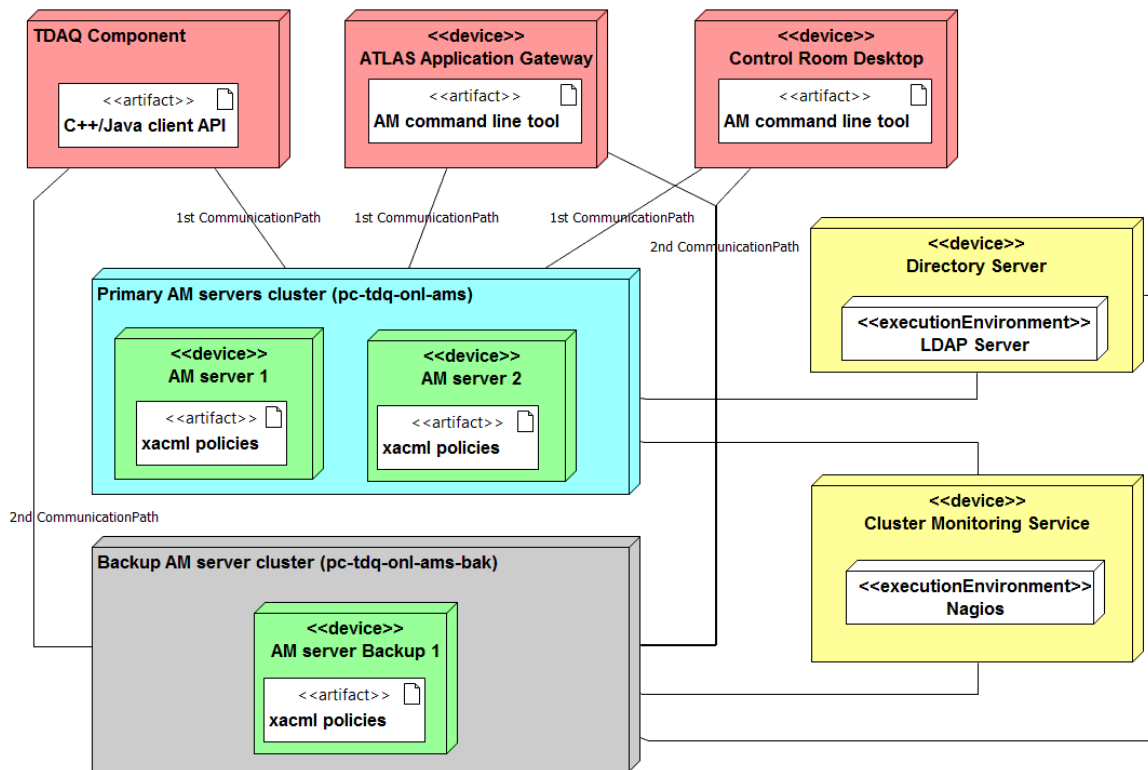


Figure 68 Production setup for TDAQ AM Service

6.5.1 AM server installation

The data used by the AM server is stored on the local disk in the `/data/AccessManager` directory in the following structure:

- `/data/AccessManager/logs` - location of the server logs
- `/data/AccessManager/policies` - location of the XACML policies used by the server
- `/data/AccessManager/service` - contains the `amServerService` script to be used to start/stop the AM server on the current machine. The directory contains a soft link to the script available in the TDAQ release installation on central file server. For example:

```
$ ls -l /data/AccessManager/service/
total 4
lrwxrwxrwx 1 mleahu zp 64 Oct 17 14:46 amServerService ->
/sw/atlas/tdaq/tdaq-01-08-03/installed/share/bin/amServerService
```

The service script reads the AM server configuration from LDAP. Figure 69 shows a sample configuration in LDAP which contains:

- Parameters to be passed by the `amServerService` script to `amServer` script which controls finally the AM server
- AM server configuration (policy set, logs folder etc)
- Server host name & service port number
- The TDAQ software version from where to start the AM server

In order to start or stop the AM server, the service script from `/etc/init.d/` (a soft link to the one in `/data/AccessManager/service`) can be used like this:

```
$ service amServerService start
```

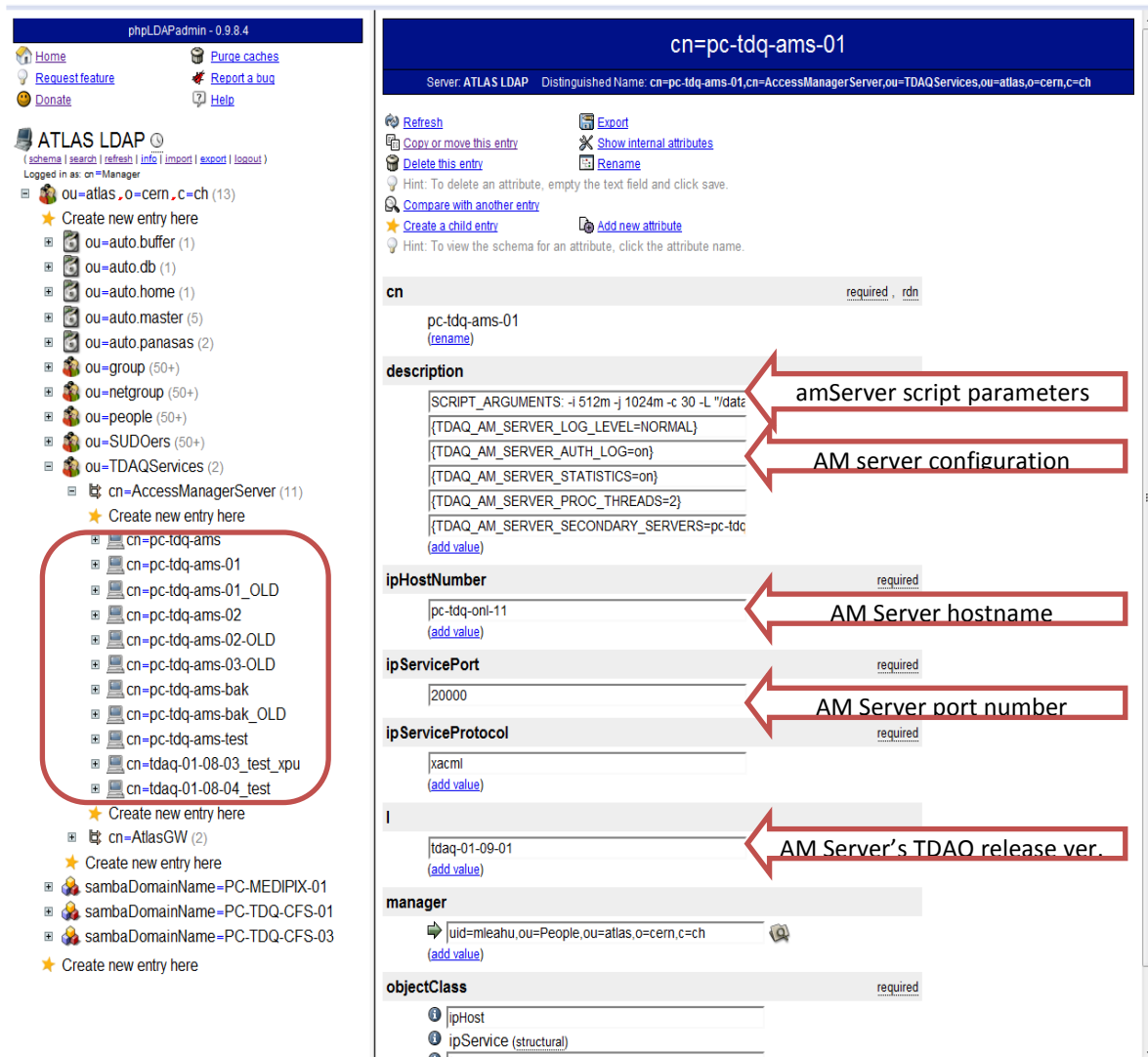


Figure 69 AM server configuration in LDAP

6.5.2 Monitoring

The AM service is integrated with the ATLAS Online cluster monitoring system - Nagios [60] through the controller interface. Nagios plugin is the AM server remote controller tool `amServerRemoteController` from chapter 6.3.2.4. The following AM server parameters are monitored:

- Server uptime
- Peak rate of requests processed OK
- Peak rate of requests processed with error
- Peak rate of requests received when the server was busy
- Average rate of requests processed OK
- Average rate of requests processed with error

Alarms are triggered and email notifications are sent to the administrators when the above parameters can't be checked or their values are out of the safe ranges. `pc-tdq-ams-*` aliases are assigned in Nagios to the cluster nodes where AM servers are hosted.

Figure 70 shows a screenshot of Nagios plots representing the history of peak processing rates for two nodes from primary cluster (`pc-tdq-ams-01` and `pc-tdq-ams-02`) and one node from backup cluster (`pc-tdq-ams-bak`). It can be observed that peak processing rates do not go above the

overload threshold (around 800 in production environment) and, when necessary, the extra load is taken by the backup server because the primary servers answered with busy status redirecting the client to the backup cluster.

The table below contains the plots for Access Manager servers.

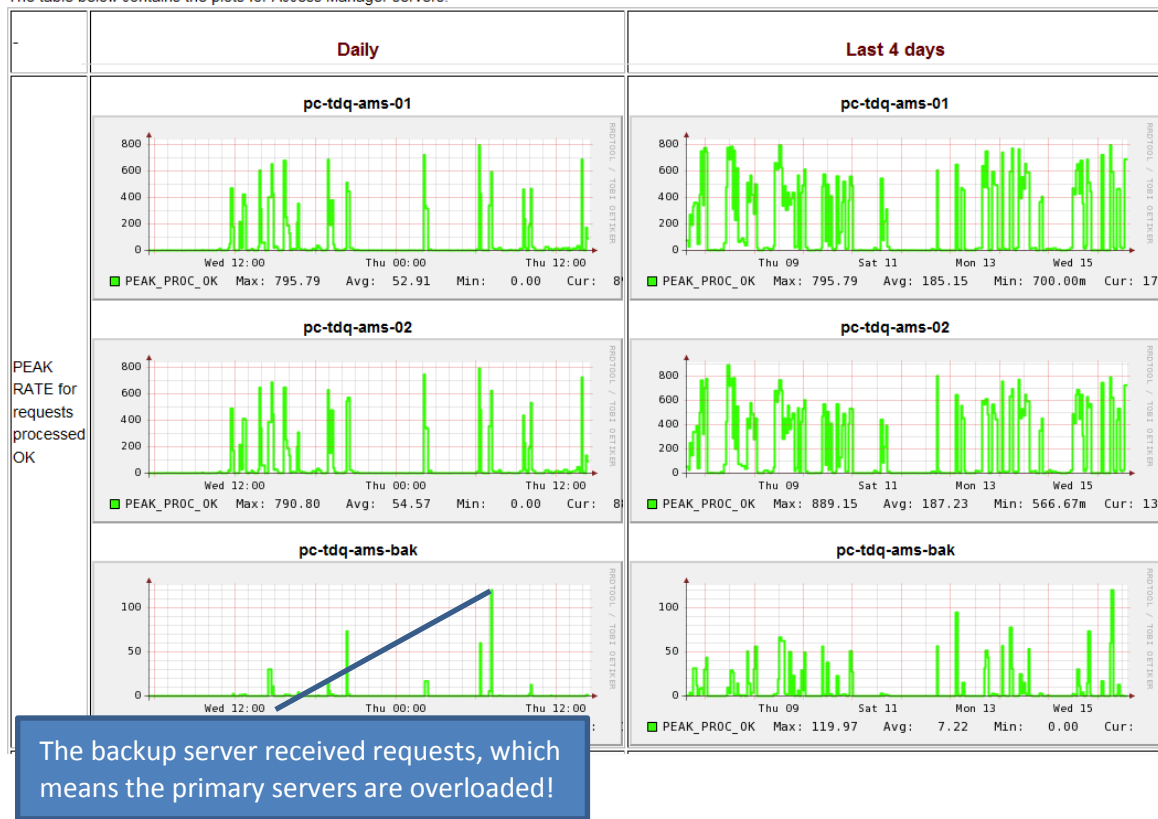


Figure 70 AM server monitoring

6.5.3 Notifications from LDAP server

The user information and roles stored in LDAP are subject to changes in time and the AM servers must be notified of these changes to clear their caches. The notification mechanism described in Figure 71 requires the following configurations and tools:

- The OpenLDAP server is configured to write any modification made to LDAP into a file using an overlay. The relevant `slapd.conf` section is:

```
database bdb
suffix "ou=atlas,o=cern,c=ch"
rootdn "cn=Manager,ou=atlas,o=cern,c=ch"
directory /usr/local/openldap/var/openldap-data

overlay auditlog
auditlog /usr/local/openldap/var/auditlog/slapd-auditlog.log
```

- `dnotify [61]` tool to monitor the file `/usr/local/openldap/var/auditlog/slapd-auditlog.log` for any change and execute the AM remote controller (chapter 6.3.2.4) tool to send a notification message to all AM servers configured in LDAP
- Helper script `amServerLDAPNotifier` prepared to control the `dnotify` tool lifecycle (start/stop, configuration):

```
$ ./amServerLDAPNotifier -h
This shell script takes care of starting and stopping the dnotify daemons to notify
the AM servers about the LDAP update events
Usage: amServerLDAPNotifier {start|stop|restart|status}
```

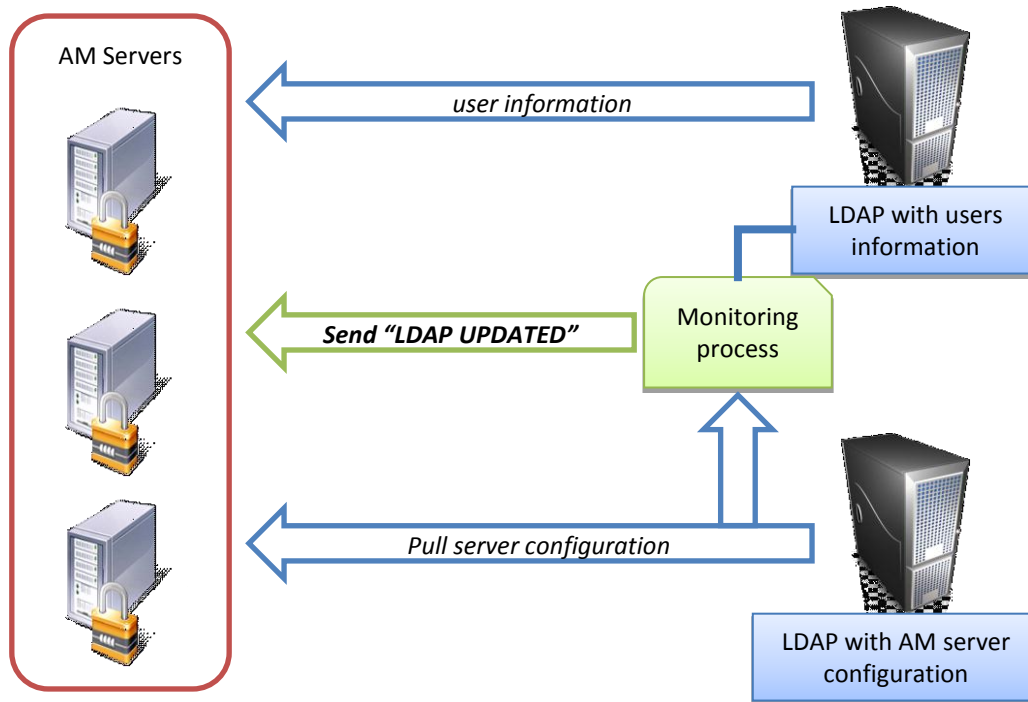


Figure 71 Notifications from LDAP servers to AM servers

6.5.4 Client configuration

The client API configuration in production setup must contain the environment variables listed in Table 21. All these variables can be hard coded in the TDAQ software configuration file `/sw/tdaq/AccessManager/cfg/client.cfg`. The configuration from this file has priority over the environment variables from a cluster node ensuring for example that the authorization is always enabled no matter the TDAQ component individual configuration on a particular node.

Table 21 Environment variables for client API

Environment variable	Values	C++ client API	Java client API
TDAQ_AM_AUTHORIZATION	"on" or "off"	required	required
TDAQ_AM_SERVER_HOST	pc-tdq-onl-ams,pc-tdq-onl-ams-bak	required	required
TDAQ_AM_SERVER_PORT	20000	required	required
TDAQ_AM_CLIENT_USE_SECONDARY_SERVERS	"true" or "false"	required	required
TDAQ_ERS_DEBUG_LEVEL	0, 1, 2 or 3	required	
TDAQ_AM_LOGS_DIR	Full path to the logs directory		required
TDAQ_AM_CLIENT_LOG_LEVEL	NONE - no logs at all NORMAL - log the errors and warnings VERBOSE - log the information messages VERY_VERBOSE - log the configuration messages DEBUG - log the debug messages ALL - log all the messages		required

Chapter 7

Conclusions

In this chapter we summarize our work on the access control system for the ATLAS experiment. We highlight the original aspects of the software solution we envisaged to protect the ATLAS Online cluster and the TDAQ software running on it. A set of topics recommended as future work on this area are outlined at the end of the chapter.

7.1 Summary of contributions

The ATLAS experiment at CERN, the largest among the other experiments on the LHC ring, represents a big step forward in physics research, but comes also with many technical challenges. The impressive amount of data generated in real time requires complex data acquisition software to filter it and a large computing cluster to run this software. The number of users foreseen to participate in the experiment supervision and data analysis is of order of thousands. Since the experiment network is required to not be completely isolated from the internet, the big computing power put in place for ATLAS is a great temptation for attackers.

All these conditions called for strengthening the security on the experiment hardware and software resources. If in previous physics experiments at CERN the actions traceability and accountability were not an issue (groups of users were using group computing accounts), the increased number of users in ATLAS made it a top priority. This concern is addressed by the access control mechanisms of the computing security area.

We performed a complete analysis, design, implementation and deployment of the access control solution for the protection of ATLAS Online cluster and the TDAQ software running on it.

The analysis phase aimed at identifying the most stringent requirements for the access control system to be implemented. The access control model suitable for experiment context was also chosen at this stage: a hierarchical Role Based Access Control model proved to satisfy the high level needs in terms of flexibility and scalability.

The design phase came with one big challenge: how to unify the OS and applications worlds under the same RBAC umbrella. The access control policies needed to be defined coherently, from the remote access to the cluster up to the very little but critical detail in the TDAQ software running to filter and collect the experiment data. We chose the LDAP directory to be the central point for definition of users, roles and, most importantly, for the assignment of users to roles. The user-roles relationships is the most dynamic part of RBAC model and, having it defined in a single place ensures that all permissions necessary for a user are available immediately once he or she gets the appropriate roles. The roles hierarchies and permissions assignment to roles have a more static nature and can be checked carefully from coherence point of view at system setup phase.

We achieved the integration of OS (SLC) with the RBAC access control using existing Linux tools and services enhanced with shell scripts and special configurations to map the RBAC concepts. Being a quasi-real-time environment, the performance of TDAQ software is crucial, so we faced the challenge of designing and implementing the most non-intrusive RBAC solution at application level to have the minimum performance impact on the TDAQ components subject to access control protection. To achieve this, the expensive task of taking the authorization decision was externalized

from the TDAQ components to a dedicated Access Manager service. This strategy brought the advantage of instant propagation of any policy change performed centrally on the AM service.

We then focused on providing a high performance AM server implementation with solutions for high availability and scalability. The high availability and load balancing are necessary to have the AM service running continually and at the optimal throughput. The scalability is mandatory given the fact that ATLAS experiment is foreseen to run for a good number of years with TDAQ software adapting to physics needs and the hardware of ATLAS Online cluster being upgraded periodically.

Other key factors of a security solution are robustness, simplicity and acceptability. We made sure to address those. In particular, well known Linux tools and services with strong security track record are part of the solution. The AM client API is customized for each TDAQ component to offer to the developers exactly what they need in terms of access control: the resource and actions are tailored for the TDAQ component's concepts, minimizing in this way the risk of AM service misuse. Last but not least the tools that allow users to request and/or grant the enabling of roles are simple to use and fast: this is a key element to ease the user's acceptability of the new "boundaries".

Before the deployment of the AM service into the production environment, we assessed thoroughly the service performance and made significant improvements. The analysis of test results provided also recommendations for the configuration of AM service in the production environment.

The current production setup consists in a high availability cluster of 6+1 nodes running the TDAQ Access Manager Service for ~3800 user accounts and ~440 roles. Each node of Access Manager Service is able to handle ~800 authorization requests per second from TDAQ software running on the ~3000 nodes of the ATLAS Online cluster. It is integrated with the system administration monitoring system for continued surveillance of service availability and performance. This production setup has run successfully in the last 4 years and has allowed ATLAS to take data steadily and efficiently, leading to the first major discovery: the Higgs boson.

7.2 Future work

During this thesis work we have achieved the goal of laying the ground work for a solid access control solution in the ATLAS Online cluster. The deployment in production confirmed the solution viability, but there is still place for extensions. The following topics can be considered for future work on the access control solution:

- The experience with access control solution during data taking might tell if the SSD/DSD constraints that we decided to omit from the model are necessary. Initial analyses didn't show it as mandatory, but the current solution can be extended with SSD/DSD constraints.
- An important and difficult task of the Administrative Component from RBAC Administration tool is to check the coherence of the overall access control policies. The lack of correlation between permission definitions, users to role assignments and SSD/DSD specifications can hide or lock system resources or actions on resources. An example of two uncorrelated access rules are the following: allow the role A to start TDAQ processes, and allow the role A to log in only to the public computers (where the TDAQ binaries are not accessible). In this case, the TDAQ process resource is hidden for the role A. Automatic validation of policies coherence would ease the task of administrators and reduce the risks of errors.
- One problem identified after using this access control solution is roles proliferation due to lack of experience and capacity of abstraction. The administrator task to keep a minimum level of consistency and simplicity is hard with the current tools. The development of a graphical tool on top of RBAC Administration tool to assist the administrator in his tasks would be welcome. The tool should offer the possibility to view the current RBAC configuration from many perspectives: to help the administrator for

example to identify duplicated roles from permissions point of view (assigned directly or by inheritance), to spot cycles in roles hierarchy, to simulate roles deletion.

- The XACML policies stored currently in files may be moved to LDAP for a single point of definition and easier propagation in case of policies update.
- The authorization logger writes now the authorizations into log files spread over the AM server nodes. Their browsing and searching is tedious, hence the integration of AM server with the TDAQ Log Server to store the authorization decisions would improve the access to the audit information.

Chapter 8

Appendices

8.1 LDAP schema for AM Roles

```
# ATLAS Role Based Access Control schema used by the Access Manager (AM)
#
#####
#
# The base OID is      : 1.3.6.1.4.1.96.64.11.0
# Attributes types are: 1.3.6.1.4.1.96.64.11.0.1
# Object classes are  : 1.3.6.1.4.1.96.64.11.0.2
#
#####
# the generic role attribute used later in the definition of other attributes
#####
attributetype ( 1.3.6.1.4.1.96.64.11.0.1.1
  NAME 'amRoleAttribute'
  DESC 'The AM role attribute'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
#####
### Attribute to specify an AM role
#####
attributetype ( 1.3.6.1.4.1.96.64.11.0.1.2
  NAME 'amRole'
  DESC 'The AM role assigned to an user'
  SUP amRoleAttribute
  SINGLE-VALUE )
#####
### Attributes needed for amRolesSet definition
#####
attributetype ( 1.3.6.1.4.1.96.64.11.0.1.3
  NAME 'amRolesSet'
  DESC 'The AM roles set assigned to an user'
  SUP amRoleAttribute
  SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.96.64.11.0.1.13
  NAME 'amRolesDescription'
  DESC 'Description of roles set'
  SUP description
  SINGLE-VALUE )

#####
### Attributes needed for role assignment and state definitions ###
#####
attributetype ( 1.3.6.1.4.1.96.64.11.0.1.4
  NAME 'amRoleAssignedBy'
  DESC 'The user id of the user who assigned this role to the current user'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
  SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.96.64.11.0.1.5
  NAME 'amRoleAssignedWhen'
  DESC 'The date when the role has been assigned to the current user'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )
```

```

attributetype ( 1.3.6.1.4.1.96.64.11.0.1.6
  NAME 'amRoleState'
  DESC 'The state of the role: enabled or disabled'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
  SINGLE-VALUE )

#####
## Attributes needed for the role definition      ##
#####
attributetype ( 1.3.6.1.4.1.96.64.11.0.1.7
  NAME 'amRoleResponsible'
  DESC 'The id of the user who created the role'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
  SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.96.64.11.0.1.8
  NAME 'amRoleJunior'
  DESC 'Role which is junior to the current role'
  SUP amRoleAttribute )

attributetype ( 1.3.6.1.4.1.96.64.11.0.1.9
  NAME 'amRoleSenior'
  DESC 'Role which is senior to the current role'
  SUP amRoleAttribute )

attributetype ( 1.3.6.1.4.1.96.64.11.0.1.10
  NAME 'amRoleSSD'
  DESC 'Role which is in an Static Separation of Duty relation with the current role'
  SUP amRoleAttribute )

attributetype ( 1.3.6.1.4.1.96.64.11.0.1.11
  NAME 'amRoleDSD'
  DESC 'Role which is in an Dynamic Separation of Duty relation with the current role'
  SUP amRoleAttribute )

attributetype ( 1.3.6.1.4.1.96.64.11.0.1.12
  NAME 'amRoleDescription'
  DESC 'Description of a role'
  SUP description
  SINGLE-VALUE )

#####
# the ATLAS roles objects
#####
objectclass ( 1.3.6.1.4.1.96.64.11.0.2.1
  NAME 'amRoles'
  SUP top
  STRUCTURAL
  DESC 'A set of ATLAS roles'
  MAY ( amRolesDescription )
  MUST ( amRolesSet ) )

# the ATLAS role reference object
objectclass ( 1.3.6.1.4.1.96.64.11.0.2.2
  NAME 'amRoleAssigned'
  DESC 'The ATLAS role'
  MUST ( amRole $ amRoleAssignedBy $ amRoleAssignedWhen $ amRoleState ) )

# the ATLAS role object
objectclass ( 1.3.6.1.4.1.96.64.11.0.2.3
  NAME 'amRoleDefinition'
  DESC 'The ATLAS role'
  MAY ( amRoleDescription $ amRoleJunior $ amRoleSenior $ amRoleSSD $ amRoleDSD )
  MUST ( amRole $ amRoleResponsible ) )

```

8.2 RBAC Administration Tool – User Requirements Document



ATLAS

ATLAS TDAQ System Administration, Controls and Coordination

ATLAS RBAC Administration Tool Requirements

Document Version: 1.0
Document ID: ATLAS-TDAQ-2007-XXX
Document Date: 16 April 2007
Document Status: Draft

Abstract

The document presents the user requirements for the ATLAS RBAC Administration Tool. They shall be the basis for the evaluation of the commercial products candidates for the ATLAS' roles and policies administration. They shall be also the basis for the design and implementation of the tool to be developed for this purpose.

Institutes and Authors:

CERN: M. Leahu

Politehnica University of Bucharest: M. Leahu

8.3 Roles Manager shell script

```
#!/bin/bash

# Access Manager script to manage the roles definition and roles hierarchies.

# Versions:
#
# 15/02/08 - mleahu
# Initial version
# 15/04/08 - mleahu
# Added -f flag for force operation
# 21/05/08 - mleahu
# Default user to bind as to LDAP is AccessManagerAdmin
# 10/06/08 - mleahu
# Fixed a bug when deleting the memberNisNetgroup attributes (all attributes were deleted
instead of one of them)

version='$Revision: 52065 $'

TRUE=1
FALSE=0
VERBOSE=$FALSE
CONFIRMATION=${TRUE}

retcode=0

function INFO()
{
    if [ "$VERBOSE" = "$TRUE" ]; then
        echo "$@"
    fi
}

function MSG()
{
    echo -n ">>> "
    echo "$@"
}

function MSGN()
{
    echo -n ">>> "
    echo -n "$@"
}

function init_grep_filter()
{
    local -a ldapattr=( $@ )
    # generate the grep filter
    grepfilter="${ldapattr[0]}:"
    for ((i=1;i<${#ldapattr[@]};i++))
    do
        grepfilter="${grepfilter}|${ldapattr[$i]}:"
    done
}

userBindDN="uid=AccessManagerAdmin,ou=People"
managerDN="cn=Manager"
passwdBind=""
ldapserver=`awk '/^host/ {printf $2}' /etc/ldap.conf`
basedn=`awk '/^base/ {printf $2}' /etc/ldap.conf`

# the LDAP specific attributes and values for roles
LDAP_ROLE_OBJECTCLASS="amRole"
LDAP_ROLE_ATTR_ASSIGNABLE="amRoleAssignable"
LDAP_ROLE_ASSIGNABLE="true"
LDAP_ROLE_NOT_ASSIGNABLE="false"
LDAP_ROLE_PROTECTED_ATTR=( "cn" "memberNisNetgroup" "nisNetgroupTriple" )
LDAP_ROLE_PREFIX_ASSIGNED="RA"
LDAP_ROLE_PREFIX_ENABLED="RE"
LDAP_ROLE_PREFIX="${LDAP_ROLE_PREFIX_ASSIGNED}"
LDAP_ROLE_SEPARATOR="-"

# the roles operations
```

```

OP_DUMP_LDIF=0
OP_SHOW=1
OP_CREATE=2
OP_DELETE=3
OP_UPDATE=4
OP_UPDATE_SENIOR_ADD=5
OP_UPDATE_SENIOR_REMOVE=6
OP_UPDATE_JUNIOR_ADD=7
OP_UPDATE_JUNIOR_REMOVE=8

ROLE_OPERATION="${OP_SHOW}"
declare -a role_update_info=( )
declare -a roles=( );

function set_role_op_value(){
    if [ "${ROLE_OPERATION}" != "${OP_SHOW}" ]
    then
        echo "Only one operation at the time allowed on the role!"
        exit 1
    fi
    ROLE_OPERATION="$1"
}

while getopts "cCdu:s:S:j:J:Lr:m:Mp:l:b:fvh" option
do
    case $option in
        h) echo "`basename $0`, version $version";
           echo "Access Manager utility to administrate roles and roles hierarchies.";
           echo "";
           echo "Usage: `basename $0` [-c|-C|-d|-u \\"(attribute=value)\\\"|[-s|-S senior]|[-j|-J
junior]|[-L] <-r ROLE> [-m userBindDN] [-M] [-p password] [-l ldapserver] [-b basedn] [-v] [-f]
[-h]";
           echo "No arguments will make the script display all the roles found in LDAP."
           echo "  -c create the role ROLE not assignable to users"
           echo "  -C create the role ROLE assignable to users"
           echo "  -d delete the role ROLE"
           echo "  -u update the ROLE attribute with a new value. The format must be
'attribute1=value1(attribute2=value2)'"
           echo "  -s add a senior role for the ROLE"
           echo "  -S remove a senior role from the ROLE"
           echo "  -j add a junior role for the ROLE"
           echo "  -J remove a junior role from the ROLE"
           echo "  -L dump the LDIF for role"
           echo "  -r the ROLE name; if no other script argument provided, the ROLE details are
displayed."
           echo "  -m authenticate as <userBindDN> to the LDAP server; default is
AccessManagerAdmin"
           echo "  -M authenticate as [${managerDN}] to the LDAP server"
           echo "  -p password to bind to the LDAP server"
           echo "  -l use this ldapserver; default is [${ldapserver}]"
           echo "  -b use this basedn; default is [${basedn}]"
           echo "  -f force mode; no confirmation asked for change operations"
           echo "  -v verbose mode"
           echo "  -h this info"
           echo ""
           echo "Author: mleahu@CERN"
           exit 0;;
        c) set_role_op_value ${OP_CREATE};role_update_info=( ${LDAP_ROLE_NOT_ASSIGNABLE} );;
        C) set_role_op_value ${OP_CREATE};role_update_info=( ${LDAP_ROLE_ASSIGNABLE} );;
        d) set_role_op_value ${OP_DELETE};;
        u) set_role_op_value ${OP_UPDATE};role_update_info=( $OPTARG );;
        s) set_role_op_value ${OP_UPDATE_SENIOR_ADD};role_update_info=( $OPTARG );;
        S) set_role_op_value ${OP_UPDATE_SENIOR_REMOVE};role_update_info=( $OPTARG );;
        j) set_role_op_value ${OP_UPDATE_JUNIOR_ADD};role_update_info=( $OPTARG );;
        J) set_role_op_value ${OP_UPDATE_JUNIOR_REMOVE};role_update_info=( $OPTARG );;
        L) set_role_op_value ${OP_DUMP_LDIF};;
        r) roles=( $OPTARG );;
        m) userBindDN="uid=${OPTARG},ou=People";;
        M) userBindDN=${managerDN};;
        p) passwdBind=$OPTARG;;
        l) ldapserver=$OPTARG;;
        b) basedn=$OPTARG;;
        f) CONFIRMATION=$FALSE;;
        v) VERBOSE=$TRUE;;
        *) echo "unknown option '$option'";;
    esac
done

```

```

shift $(( $OPTIND - 1 ))

userBindDN="{userBindDN},$basedn"
rolesBaseDN="ou=netgroup,$basedn"

if ( ! which ldapsearch > /dev/null 2>&1 )
then
    echo "ldapsearch tool not found!"
    exit 2
fi

ldapoptions="-h ${ldapserver} -b ${rolesBaseDN} -x"
# sort the output based on cn
ldapoptions="${ldapoptions} -S cn -LLL"
ldapfilter="(objectClass=${LDAP_ROLE_OBJECTCLASS})"
ldapwriteoptions=" -W "

if [ "${ROLE_OPERATION}" -gt "${OP_SHOW}" ]; then
    if [ "${#roles[@]}" -eq "0" ]; then
        echo "No role specified! Abort!"
        exit 1
    fi

    # get the password to bind without
    if [ -z "${passwdBind}" ]
    then
        echo -e "Password for ${userBindDN}:"
        read -s passwdBind
        if [ -z "${passwdBind}" ] ;
        then
            echo "Empty password for ${userBindDN} account!"
            exit 1
        fi
    fi
    ldapwriteoptions=" -w ${passwdBind} "
fi

if [ "${VERBOSE}" = "${TRUE}" ]; then
    ldapwriteoptions="${ldapwriteoptions} -v"
fi

function if_role_exists()
{
    local _role=${1}
    local _options="${ldapoptions}"
    local _filter="( &${ldapfilter} (cn=${LDAP_ROLE_PREFIX}${LDAP_ROLE_SEPARATOR}${_role}))"
    local -a _attr=( "cn" )
    ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep -q ${_role}
}

#fill in the role_seniors array
function get_role_seniors() {
    local _role=${1}
    local _options="${ldapoptions}"
    local _filter="( &${ldapfilter} (cn=${LDAP_ROLE_PREFIX}${LDAP_ROLE_SEPARATOR}${_role}))"
    local -a _attr=( "memberNisNetgroup" )
    init grep filter ${_attr[@]}
    role_seniors=( `ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep -P ${grepfilter} |
cut -d '-' -f 2- | sort | uniq` )
}

#fill in the role_juniors array
function get_role_juniors()
{
    local _role=${1}
    local _options="${ldapoptions}"
    local
    _filter="( &${ldapfilter} (memberNisNetgroup=${LDAP_ROLE_PREFIX}${LDAP_ROLE_SEPARATOR}${_role}))"
    "
    local -a _attr=( "cn" )
    init grep filter ${_attr[@]}
    role_juniors=( `ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep -P ${grepfilter} |
cut -d '-' -f 2- | sort | uniq` )
}

if [ "${ROLE_OPERATION}" = "${OP_DUMP_LDIF}" ]; then

```



```

#####
#INFO "DUMP LDIF"
#####
filter=""
for role in ${roles[@]}
do
    if [ -z "${filter}" ]; then
        filter="(cn=${LDAP_ROLE_SEPARATOR}${role})"
    else
        filter="(|(cn=${LDAP_ROLE_SEPARATOR}${role})${filter})"
    fi
done
if [ -z "${filter}" ]; then
    filter="${ldapfilter}"
else
    filter="(&${ldapfilter}${filter})"
fi
ldapsearch ${ldapoptions} "${filter}"

elif [ "${ROLE_OPERATION}" = "${OP_SHOW}" ]; then

#####
INFO "DISPLAY THE ROLE INFORMATION"
#####

function show_role(){
    local _role=$1
    local _options="${ldapoptions}"
    local _filter="(&${ldapfilter})(cn=${LDAP_ROLE_PREFIX}${LDAP_ROLE_SEPARATOR}${_role})"
    local -a _attr=( "cn" )

    #display role properties
    init_grep_filter ${_attr[@]}
    local -a result=( `ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep -P
${grepfilter} | sort | cut -d '-' -f2-` )

    if [ "${#result[@]}" -gt "1" ]; then
        echo "### Found more roles with name [$role]: ${#result[@]} roles #####"
    fi

    _attr=( "amRoleAssignable" )
    init_grep_filter ${_attr[@]}

    for _role_in_result in ${result[@]}
    do
        echo "##### ROLE [${_role_in_result}] #####"

        _filter="(&${ldapfilter})(cn=${LDAP_ROLE_PREFIX}${LDAP_ROLE_SEPARATOR}${_role_in_result})"
        init_grep_filter ${_attr[@]}
        ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep -P ${grepfilter} | sort

        # display role juniors:
        get_role_juniors ${_role_in_result}
        echo "JUNIORS(${#role_juniors[@]} ${role_juniors[@]})"

        # display role seniors:
        get_role_seniors ${_role_in_result}
        echo "SENIORS(${#role_seniors[@]} ${role_seniors[@]})"

        echo
    done
}

if [ -z "${roles}" ]; then
    #get all the roles
    filter="(&${ldapfilter})(cn=${LDAP_ROLE_PREFIX}${LDAP_ROLE_SEPARATOR}*)"
    attr=( cn )
    init_grep_filter ${attr[@]}
    all_roles=( `ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter}
| cut -d '-' -f 2- | uniq` )
    if [ "${VERBOSE}" = "${TRUE}" ]; then
        echo "### ${#all_roles[@]} ROLES IN LDAP [ ${all_roles[@]} ]"
        echo "#----- DETAILS -----#"
        # display the details of all the roles
        for role in ${all_roles[@]}

```

```

do
    show_role ${role}
done
else
    #display all the roles
    echo ${all_roles[@]} | tr ' ' '\n' | sort
fi
else
    for role in ${roles[@]}
    do
        show_role ${role}
    done
fi

elif [ "${ROLE_OPERATION}" = "${OP_CREATE}" ]; then
#####
INFO "CREATE ROLE"
#####

for role in ${roles[@]}
do
    ldap_amRoleAssignable="${role_update_info[@]}"
    msg_roleAssignable="ASSIGNABLE"
    if [ "${ldap_amRoleAssignable}" = "${LDAP_ROLE_NOT_ASSIGNABLE}" ]; then
        msg_roleAssignable="NOT ASSIGNABLE"
    fi

    MSGN "Create ${msg_roleAssignable} role [$role]..."
    if ( ! if_role_exists ${role} ) ; then
        ldap_cn="${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}"
        ldap_dn="cn=${ldap_cn},${rolesBasedN}"
        # create the role
        ldap_ldif="dn: ${ldap_dn}
objectClass: top
objectClass: nisNetgroup
objectClass: ${LDAP_ROLE_OBJECTCLASS}
cn: ${ldap_cn}
amRoleAssignable: ${ldap_amRoleAssignable}
description: The ATLAS role [$role] ${msg_roleAssignable} to users!"

        # create the role enabled netgroup as well
        ldap_cn="${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role}"
        ldap_dn="cn=${ldap_cn},${rolesBasedN}"
        ldap_ldif="${ldap_ldif}
dn: ${ldap_dn}
objectClass: top
objectClass: nisNetgroup
objectClass: ${LDAP_ROLE_OBJECTCLASS}
cn: ${ldap_cn}
amRoleAssignable: ${ldap_amRoleAssignable}
description: The ATLAS role [$role] ${msg_roleAssignable} to users in ENABLED state!"

        echo "${ldap_ldif}" | ldapadd -x ${ldapwriteoptions} -D "${userBindDN}" -h
${ldapserver} > /dev/null
        if [ "${PIPESTATUS[1]}" != "0" ]; then
            MSG "OPERATION FAILED! Role NOT created!"
            INFO "LDIF: ${ldap_ldif}"
        else
            MSG "DONE!"
        fi
    else
        echo "SKIPPED! Already exists!"
    fi
done

exit 0

elif [ "${ROLE_OPERATION}" = "${OP_DELETE}" ]; then
#####
INFO "DELETE ROLES"
#####

if [ "${CONFIRMATION}" = "${TRUE}" ]; then
    echo "WARNING: FOLLOWING ${#roles[@]} ATLAS ROLES WILL BE PERMANENTLY DELETED FROM
LDAP:"

```

```

echo "${roles[@]}"
read -p "ARE YOU SURE (y/n)?" -n 1 answer
echo
if [ "${answer}" != "y" ]; then
echo "Operation aborted!"
exit 2
fi

fi

for role in ${roles[@]}
do
MSGN "Delete role [$role]..."
_options=${ldapoptions}

_filter="( (&${ldapfilter})(|(cn=${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role})(cn=${
LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role})) )"
_attr=( "cn" )
role_dns=`ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep '^dn:' | cut -d ':'
-f2-`

if [ -n "${role_dns}" ] ; then
echo "${role_dns}" | ldapdelete -x ${ldapwriteoptions} -D "${userBindDN}" -h
${ldapserver}

if [ "${PIPESTATUS[1]}" != "0" ]; then
MSG "OPERATION FAILED! Role NOT deleted!"
else
MSG "DONE!"
fi

MSG "Delete junior-senior relations..."

_filter="( (&${ldapfilter})(|(memberNisNetgroup=${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}
}${role})(memberNisNetgroup=${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role})) )"
_attr=( "memberNisNetgroup" )

_grep_filter="^dn|${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}|${LDAP_ROLE_PREFIX
ENABLED}${LDAP_ROLE_SEPARATOR}${role}|^$"
ldap_ldif=`ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep -E
"${_grep_filter}" | while read line ; do echo $line;(echo ${line} | grep -q '^dn:') && (echo
"delete: ${_attr[0]}");done`
echo "${_ldap_ldif}" | ldapmodify -x ${ldapwriteoptions} -D "${userBindDN}" -h
${ldapserver}

if [ "${PIPESTATUS[1]}" != "0" ]; then
MSG "OPERATION FAILED! Relationships NOT deleted!"
INFO "LDIF: ${_ldap_ldif}"
else
MSG "DONE!"
fi
else
echo "SKIPPED! Not found!"
fi
done

elif [ "${ROLE_OPERATION}" = "${OP_UPDATE}" ] ; then
#####
INFO "UPDATE THE ROLE INFORMATION"
#####
if [ "${#role_update_info[@]}" -eq 0 ] ; then
echo "No role information provided!"
exit 3
fi

for role in ${roles[@]}
do
MSGN "Update information for role [$role]..."
if ( ! if_role_exists ${role} ) ; then
echo "SKIPPED! Role not found!"
continue
fi
echo
ldap_ldif=""
for info in `echo "${role_update_info[@]}" | tr -d '(' | tr ')' '\n'`
do
MSGN "...[${info}]..."
attr_name="${info%#*}"
attr_value="${info#*#*}"
change_allowed="${TRUE}"

```

```

for attr_prot in ${LDAP_ROLE_PROTECTED_ATTR[@]}
do
    if [ "${attr_prot}" = "${attr_name}" ]; then
        change_allowed="${FALSE}"
        break;
    fi
done

if [ "${change_allowed}" = "${FALSE}" ]; then
    echo "PROTECTED ATTRIBUTE!"
    continue;
fi
echo

ldap_ldif="${ldap_ldif}dn:
cn=${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role},${rolesBasedN}
replace: ${attr_name}
${attr_name}: ${attr_value}

dn: cn=${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role},${rolesBasedN}
replace: ${attr_name}
${attr_name}: ${attr_value}

"

done
echo "${ldap_ldif}" | ldapmodify -x ${ldapwriteoptions} -D "${userBindDN}" -h
${ldapserver}
if [ "${PIPESTATUS[1]}" != "0" ]; then
    MSG "FAILED! Information not updated for ${role}!"
    INFO "LDIF: ${ldap_ldif}"
else
    MSG "DONE!"
fi
done
elif [ "${ROLE_OPERATION}" = "${OP_UPDATE_JUNIOR_ADD}" -o "${ROLE_OPERATION}" =
"${OP_UPDATE_SENIOR_ADD}" -o "${ROLE_OPERATION}" = "${OP_UPDATE_JUNIOR_REMOVE}" -o
"${ROLE_OPERATION}" = "${OP_UPDATE_SENIOR_REMOVE}" ]; then
    #####
    INFO "PROCESS JUNIOR-SENIOR ROLES RELATIONSHIPS"
    #####
    if [ "${ROLE_OPERATION}" = "${OP_UPDATE_JUNIOR_ADD}" -o "${ROLE_OPERATION}" =
"${OP_UPDATE_JUNIOR_REMOVE}" ]; then
        relation_type="junior"
    else
        relation_type="senior"
    fi

    if [ "${ROLE_OPERATION}" = "${OP_UPDATE_JUNIOR_ADD}" -o "${ROLE_OPERATION}" =
"${OP_UPDATE_SENIOR_ADD}" ]; then
        relation_op="add"
    else
        relation_op="delete"
    fi

    if [ "${#role_update_info[@]}" -eq 0 ]; then
        echo "No ${relation_type} roles provided!"
        exit 3
    fi

for role in ${roles[@]}
do
    MSGN "${relation_op} ${relation_type}s for role [${role}]..."
    if ( ! if_role_exists ${role} ); then
        echo "SKIPPED! Role not found!"
        continue
    fi
    echo
    ldap_ldif=""
    for xnior in ${role_update_info[@]}
    do
        echo -n "...${relation_type} [${xnior}]:"
        if ( ! if_role_exists ${xnior} ); then
            echo "SKIPPED! Role not found!"
            continue
        else
            echo "OK"
        fi
    done

```

```

        if [ "${relation_type}" = "junior" ]; then
            # the junior of a role contains a memberNisNetgroup attribute with the role
value
            ldap_ldif="${ldap_ldif}dn:
cn=${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${xnior},${rolesBaseDN}
${relation_op}: memberNisNetgroup
memberNisNetgroup: ${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}

dn: cn=${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${xnior},${rolesBaseDN}
${relation_op}: memberNisNetgroup
memberNisNetgroup: ${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role}

"

            else
                # the senior of a role is a memberNisNetgroup value in the role netgroup
definition
                ldap_ldif="${ldap_ldif}dn:
cn=${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role},${rolesBaseDN}
${relation_op}: memberNisNetgroup
memberNisNetgroup: ${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${xnior}

dn: cn=${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role},${rolesBaseDN}
${relation_op}: memberNisNetgroup
memberNisNetgroup: ${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${xnior}

"

            fi
        done
        echo "${ldap_ldif}" | ldapmodify -x ${ldapwriteoptions} -D "${userBindDN}" -h
${ldapservers}
        if [ "${PIPESTATUS[1]}" != "0" ]; then
            MSG "FAILED! Relationships not changed for ${role}!"
            INFO "LDIF: ${ldap_ldif}"
        else
            MSG "DONE!"
        fi
    done
else
    echo "Internal error: unknown role operation code ${ROLE_OPERATION}";
fi

```

8.4 Roles Display PHP script

LDAP_roles.php: helper script that retrieves the roles from LDAP.

```

<?php

/**
 * ATLAS Point 1 roles retrieved from LDAP.
 *
 * @author Marius Leahu
 */

if (version_compare(PHP_VERSION(), '5.0.0', '<'))
{
    die('You need at least PHP version 5.0.0!');
}

class LDAP_roles
{
    // keys for the roles array
    const ROLE_ASSIGNABLE = 'assignable';
    const NODE_ID = 'nodeid';
    const PARENT_NODE_IDS_ARRAY = 'parentnodeids';
    const SENIORS_ARRAY = 'seniors';
    const JUNIORS_ARRAY = 'juniors';
    const MEMBERS_ASSIGNED_ARRAY = 'membersassigned';
    const MEMBERS_ENABLED_ARRAY = 'membersenabled';

    const LDAP_ROLE_PREFIX_ASSIGNED = 'RA-';
    const LDAP_ROLE_PREFIX_ENABLED = 'RE-';
    const LDAP_OBJ_ROLE = 'amRole';
    const LDAP_ATTR_ROLE_NAME = 'cn';

```

```

const LDAP_ATTR_ROLE_ASSIGNABLE = 'amroleassignable';
const LDAP_ATTR_MEMBERS = 'nisnetgrouptriple';
const LDAP_ATTR_SENIORS = 'membernisnetgroup';

private $ldap_host;
private $ldap_basedn;
private $roles = array();

private $verboseMode;

public function __construct($ldap_host, $ldap_basedn) {
    $this->ldap_host = $ldap_host;
    $this->ldap_basedn = $ldap_basedn;
    $this->verboseMode = false;
}

public function setVerboseMode($verboseMode) {
    $this->verboseMode = $verboseMode;
}

private function print_msg($msg) {
    if ( $this->verboseMode ) echo $msg;
}

private function print_r_msg($array) {
    if ( $this->verboseMode ) print_r($array);
}

/**
 * retrieves the roles from LDAP
 * @param string $ldap_search_filter
 * @return array with search results as returned by the ldap search function
 */
private function searchLDAPforRoles($ldap_search_filter) {
    $con=ldap_connect($this->ldap_host);
    if ($con) {
        ldap_set_option($con, LDAP_OPT_PROTOCOL_VERSION, 3);

        $ldapb=ldap_bind($con); // this is an "anonymous" bind, typically read-only access
        if ($ldapb){
            $sr=ldap_search($con, $this->ldap_basedn, $ldap_search_filter);
            ldap_sort($con, $sr, self::LDAP_ATTR_ROLE_NAME);
            $info = ldap_get_entries($con, $sr);
            return $info;
        } else {
            echo "<h4>Unable to bind to LDAP server ". $this->ldap_host ." </h4>";
        }
        ldap_close($con);
    } else {
        echo "<h4>Unable to connect to LDAP server</h4>";
    }
}

/**
 * get the roles in an array
 * @param int $role_prefix optionally parameter to get only the role with a given prefix
 * @return array with the following structure: for each rolename, there are:
 *     the boolean attribute $roles[rolename][assignable]
 *     the node id $roles[rolename][nodeid]
 *     the parent node id $roles[rolename][parentnodeid]
 *     the users member of this role $roles[rolename][users]
 *     the roles seniors and juniors: $roles[rolename][seniors],
 *     $roles[rolename][juniors]
 */
public function getRoles($role_prefix="") {
    $pos = strpos($role_prefix, "!");
    if ( $pos !== false && $pos == 0 ) {
        $role_prefix = substr($role_prefix, 1);
    }
    $role_name_filter = "(|" .
        "(". self::LDAP_ATTR_ROLE_NAME ."=". self::LDAP_ROLE_PREFIX_ASSIGNED .
    $role_prefix ."*)".
        "(". self::LDAP_ATTR_ROLE_NAME ."=". self::LDAP_ROLE_PREFIX_ENABLED .
    $role_prefix ."*)".
        ")";
    if ( $pos !== false && $pos == 0 ) {
        $role_name_filter = "&".
            "(!". $role_name_filter .")".

```

```

        "(" . self::LDAP_ATTR_ROLE_NAME ."=" . self::LDAP_ROLE_PREFIX_ASSIGNED
        ."*")".
        "(" . self::LDAP_ATTR_ROLE_NAME ."=" . self::LDAP_ROLE_PREFIX_ENABLED
        ."*")".
        ")"".
        ")";
    }
    $ldap_search_filter = "((&.$role_name_filter.(objectClass=. self::LDAP_OBJ_ROLE .
    ))";
    $this->print_msg("LDAP search filter [".$ldap_search_filter."]);

    $ldap_search_result=$this->searchLDAPforRoles($ldap_search_filter);
    $this->print_r msg($ldap_search_result);

    $this->roles = array();
    // iterate first time to extract the role names from the search result
    for($i=0; $i<$ldap_search_result['count']; $i++) {
        // get the role name from LDAP search result
        $role_name = $ldap_search_result[$i][self::LDAP_ATTR_ROLE_NAME][0];

        $pos = strpos($role_name, self::LDAP_ROLE_PREFIX_ASSIGNED);
        // check if the role name has the ASSIGNED prefix at position 0
        if ($pos === false || $pos != 0 ) continue;

        //cut both RA and RE prefixes
        $role_name = substr($role_name, strlen(self::LDAP_ROLE_PREFIX_ASSIGNED));

        // initialize the attributes
        $this->roles[$role_name][self::ROLE_ASSIGNABLE] = "";
        $this->roles[$role_name][self::NODE_ID] = $i + 1;
        $this->roles[$role_name][self::PARENT_NODE_IDS_ARRAY] = array();
        $this->roles[$role_name][self::MEMBERS_ASSIGNED_ARRAY] = array();
        $this->roles[$role_name][self::MEMBERS_ENABLED_ARRAY] = array();
        $this->roles[$role_name][self::SENIORS_ARRAY] = array();
        $this->roles[$role_name][self::JUNIORS_ARRAY] = array();
    }

    $current_node_id = $ldap_search_result['count'] + 1;

    // set the value of the attributes
    for($i=0; $i<$ldap_search_result['count']; $i++) {
        $ldap_role = $ldap_search_result[$i];

        // get the role name from LDAP search result
        $role_name = $ldap_role[self::LDAP_ATTR_ROLE_NAME][0];
        $role_type_prefix = self::LDAP_ROLE_PREFIX_ASSIGNED;

        $pos = strpos($role_name, self::LDAP_ROLE_PREFIX_ENABLED);
        // check if the role name has the ENABLED prefix at position 0
        if ($pos !== false && $pos == 0 ) {
            $role_type_prefix = self::LDAP_ROLE_PREFIX_ENABLED;
        }
        // cut the prefix
        $role_name = substr($role_name, strlen($role_type_prefix));

        // set the assignable attribute
        $this->roles[$role_name][self::ROLE_ASSIGNABLE] = false;
        if (array_key_exists(self::LDAP_ATTR_ROLE_ASSIGNABLE, $ldap_role)) {
            $this->roles[$role_name][self::ROLE_ASSIGNABLE] =
            $ldap_role[self::LDAP_ATTR_ROLE_ASSIGNABLE][0];
        } else {
            echo "ERROR: Attribute [" . self::LDAP_ATTR_ROLE_ASSIGNABLE ."] not found for role
            [" . $role_name ."]";
        }

        //add the users
        if (array_key_exists(self::LDAP_ATTR_MEMBERS, $ldap_role)) {
            for($j=0; $j<$ldap_role[self::LDAP_ATTR_MEMBERS]['count']; $j++) {
                if ( $role_type_prefix == self::LDAP_ROLE_PREFIX_ASSIGNED)
                    $this->roles[$role_name][self::MEMBERS_ASSIGNED_ARRAY][$j] =
                    str_replace(array("(", ",", ")", " "), "", $ldap_role[self::LDAP_ATTR_MEMBERS][$j]);
                if ( $role_type_prefix == self::LDAP_ROLE_PREFIX_ENABLED)
                    $this->roles[$role_name][self::MEMBERS_ENABLED_ARRAY][$j] =
                    str_replace(array("(", ",", ")", " "), "", $ldap_role[self::LDAP_ATTR_MEMBERS][$j]);
            }
        }
    }
}

```

```

// process the seniors and juniors only for the roles assigned
if ( $role_type_prefix == self::LDAP_ROLE_PREFIX_ENABLED) continue;

//add the senior roles of this role, then this role as junior of the senior
if (array_key_exists(self::LDAP_ATTR_SENIORS, $ldap_role)) {
    for($j=0; $j<$ldap_role[self::LDAP_ATTR_SENIORS]['count']; $j++) {
        // get the senior name
        $senior_name = $ldap_role[self::LDAP_ATTR_SENIORS][$j];
        $senior_name = substr($senior_name, strlen($role_type_prefix));

        // check if the senior is still part of the roles in the search result; if
not, then skip
        if ( ! array_key_exists($senior_name, $this->roles) ) continue;

        // add it to the list of seniors
        $this->roles[$role_name][self::SENIORS_ARRAY][$j] = $senior_name;

        // add the junior relation
        $this->roles[$senior_name][self::JUNIORS_ARRAY][] = $role_name;
    }
}
return $this->roles;
}
}
?>

```

roles_img.php: the script that takes the data from LDAP_roles.php and feeds it to the GDRenderer.php [43] script.

```

<?php
/**
 * ATLAS Point 1 roles retrieved from LDAP.
 * Uses: phpTreeGraph
 *
 * @author Marius Leahu
 */

// get the parameters
$ldap_host = "localhost";
$ldap_role_prefix = "";
$top_role_filter = "LAR";
$test_mode = "";

if ($test_mode == "") {
    $ldap_host = $_GET['ldap_host'];
    $ldap_role_prefix = $_GET['ldap_role_prefix'];
    $top_role_filter = $_GET['top_role_filter'];
}

if ( $ldap_host == "" ){
    $ldap_host = "localhost";
}

function print_test($msg){
    if ( $test_mode != "" ) echo $msg;
}

//include GD rendering class
require_once('./classes/GDRenderer.php');

//create new GD renderer, optional parameters: LevelSeparation, SiblingSeparation,
SubtreeSeparation, defaultNodeWidth, defaultNodeHeight
$objTree = new GDRenderer(35, 10, 120);

//include LDAP roles class
require_once('./LDAP_roles.php');

$roles_obj = new LDAP_roles($ldap_host, "ou=atlas,o=cern,c=ch");

```



```

$roles = $roles_obj->getRoles($ldap_role_prefix);

//add nodes to the tree, parameters: id, parentid optional text, width, height, image(path)
$current_roles_count=count($roles);
$previous_roles_count=$current_roles_count + 1;
$current_node_id = 0;
while ($current_roles_count < $previous_roles_count ){
    $previous_roles_count=$current_roles_count;

    foreach($roles as $role_name => $role){
        // skip the node if it has some seniors because its seniors must be processed first
        if (count($role[LDAP_roles::SENIORS_ARRAY]) > 0) continue;

        if (count($role[LDAP_roles::PARENT_NODE_IDS_ARRAY]) == 0) {
            $role[LDAP_roles::PARENT_NODE_IDS_ARRAY][] = 0;

            // this is a top level role, so let's see if it matches the filter
            if ( $top_role_filter != "" && ! preg_match("/" . $top_role_filter . "/",
            $role_name) ) {

                // this role doesn't match the filter, so let's remove it and then delete it
                from children's seniors list
                unset($roles[$role_name]);
                foreach($role[LDAP_roles::JUNIORS_ARRAY] as $junior) {
                    // remove the current role from the its junior's seniors array
                    foreach($roles[$junior][LDAP_roles::SENIORS_ARRAY] as $j => $jsenior_name){
                        if ($jsenior_name == $role_name) {
                            unset($roles[$junior][LDAP_roles::SENIORS_ARRAY][$j]);
                        }
                    }
                }
                // continue with the next role
                continue;
            }
        }

        // now, for each parent node id, create a node object with new id and add each of this
        ids to the juniors' parent node ids array
        foreach($role[LDAP_roles::PARENT_NODE_IDS_ARRAY] as $parent_node_id) {
            // create the node object
            $current_node_id++;
            $img_name = "roles na.png";
            $node_text = $role_name;

            if ($role[LDAP_roles::ROLE_ASSIGNABLE] == 'true'){
                $img_name = "roles a.png";
                $node_text = $node_text . " " .
                "(".
                count($role[LDAP_roles::MEMBERS_ENABLED_ARRAY]) . "/" .
                count($role[LDAP_roles::MEMBERS_ASSIGNED_ARRAY]) .
                ")";
            }
            $node_width = strlen($node_text) * 8 + 4;
            $node_width = max($node_width, 100);
            $objTree->add($current_node_id, $parent_node_id, $node_text, $node_width, 20,
            $img_name);
            foreach($role[LDAP_roles::JUNIORS_ARRAY] as $junior) {
                $roles[$junior][LDAP_roles::PARENT_NODE_IDS_ARRAY][] = $current_node_id;
                // remove the current role from the its junior's seniors array
                foreach($roles[$junior][LDAP_roles::SENIORS_ARRAY] as $j => $jsenior_name){
                    if ($jsenior_name == $role_name)
                        unset($roles[$junior][LDAP_roles::SENIORS_ARRAY][$j]);
                }
            }
            unset($roles[$role_name]);
        }
        $current_roles_count=count($roles);
    }
}

if ($current_roles_count > 0) {
    echo "ERROR: Cycle found!";
    print_r($roles);
    return 1;
}

$objTree->setBGColor(array(255, 255, 255));

```

```
$objTree->setNodeColor(array(0, 128, 255));
$objTree->setLinkColor(array(0, 64, 128));
$objTree->setNodeBorder(array(0, 0, 0), 2);
$objTree->setFTFont('/usr/share/fonts/bitstream-vera/Ver.ttf', 10, 0, GDRenderer::CENTER |
GDRenderer::TOP);

if ($test_mode == "") {
    $objTree->stream();
}

?>
```

8.5 User's roles management shell script

```
#!/bin/bash

# Access Manager script to manage the users-roles. relationship.

# Versions:
#
# 18/02/08 - mleahu
# Initial version
# 22/05/08 - mleahu
# Default user to bind as to LDAP is AccessManagerAdmin
# 07/08/08 - mleahu
# Fixed a bug in the MSG(N) functions
# 04/09/08 - mleahu
# Added -G flag to specify a role category to prefix the roles to be processed
# Allow to process more users in the same time provided either in the command line or in a
input text file
# 05/09/08 - mleahu
# Added -L flag to display the roles defined in LDAP
# 08/09/08 - mleahu
# Convert usernames to lower case
# 10/10/12 - mleahu
# Added -D option to show the roles assigned and disabled
# Added -Q option to show the users with a role assigned and disabled

version='$Revision: 52762 $'

TRUE=1
FALSE=0
VERBOSE=$FALSE

retcode=0

function INFO()
{
    if [ "$VERBOSE" = "$TRUE" ]; then
        echo "$@"
    fi
}

function MSG()
{
    echo -n ">>> "
    echo "$@"
}

function MSGN()
{
    echo -n ">>> "
    echo -n "$@"
}

function init_grep_filter()
{
    local -a ldapattr=( $@ )
    # generate the grep filter
    grepfilter="${ldapattr[0]}:"
    for ((i=1;i<${#ldapattr[@]};i++))
    do
        grepfilter="${grepfilter}|${ldapattr[$i]}:"
    done
}
```

```

done
}

user_default=`whoami`
managerDN="cn=Manager"
#userBindDN="uid=AccessManagerAdmin,ou=People"
userBindDN=${managerDN}
passwdBind="marius"
ldapservers=`awk '/^host/ {printf $2}' /etc/ldap.conf`
basedn=`awk '/^base/ {printf $2}' /etc/ldap.conf`

# the LDAP specific attributes and values for roles
LDAP_ROLE_OBJECTCLASS="amRole"
LDAP_ROLE_ATTR_ASSIGNABLE="amRoleAssignable"
LDAP_ROLE_ASSIGNABLE="true"
LDAP_ROLE_NOT_ASSIGNABLE="false"
LDAP_ROLE_PREFIX_ASSIGNED="RA"
LDAP_ROLE_PREFIX_ENABLED="RE"
LDAP_ROLE_PREFIX="{LDAP_ROLE_PREFIX_ASSIGNED}"
LDAP_ROLE_SEPARATOR="-"
LDAP_USER_ROLE_ATTR="nisNetgroupTriple"

function format_user_for_ldap_filter()
{
    local _user=${1}
    echo "\\ (,${user},\\" )"
}

function format_user_for_role()
{
    local _user=${1}
    echo "(,${user},)"
}

# the roles operations
OP_HELP=0
OP_SHOW_USER_ROLES=1
OP_SHOW_ROLES=2
OP_SHOW_ROLES_CONTENT=3
OP_ASSIGN=4
OP_ASSIGN_ENABLE=5
OP_REVOKE=6
OP_ENABLE=7
OP_DISABLE=8

ROLE_STATE_ASSIGNED="assigned"
ROLE_STATE_ENABLED="enabled"
ROLE_STATE_DISABLED="disabled"
role_state="";
output_for_file="{FALSE}"
input_file=""

role_category=""

OP_SHOWS=${OP_SHOW_ROLES_CONTENT}
OPERATION="{OP_HELP}"
declare -a roles=( );
declare -a users=( );

function set_op_value(){
    if [ "${OPERATION}" -gt "${OP_HELP}" ]
    then
        echo "Only one operation at the time is allowed!"
        exit 1
    fi
    OPERATION="$1"
}

while getopts "hG:La:A:r:e:d:c:C:Q:sSdfu:F:m:Mp:l:b:v" option
do
    case $option in
        h) echo "`basename $0`, version $version";
           echo "Access Manager script to manage the users-roles RBAC relationship.";
           echo ""
           echo "-n Usage: `basename $0` [-h] [-G role_category] [-L] [-a|-A|-r|-e|-d|-c|-C|-Q
roles] [-s|-S|-D] [-f] [-u usernames] [-F input_file]";
           if [ -z "${role_category}" ]; then

```

```

# the user is not restricted to a category,
echo -n " [-m userBindDN] [-M] [-p password] [-l ldapserver] [-b basedn] [-v]";
fi
echo ""
echo " -h this info"
echo ""
echo " -G <role_category> to be used as prefix for the <roles>;"
if [ -n "${role_category}" ]; then
echo "     EXAMPLE: the role_category ${role_category} and the roles 'shifter
DAQ:expert' "
echo "     will produce the roles '${role_category}:shifter
${role_category}:DAQ:expert' "
echo "     which will be processed according to the other flags"
fi
echo ""
echo " Roles operations (only ONE operation from the list below can be called):"
echo " -L list the roles defined in LDAP in the role_category (if specified)"
echo " -a assign the <roles> to the <username>"
echo " -A assign and enable the <roles> to the <username>"
echo " -r revoke the <roles> for the <username>"
echo " -e enable the <roles> for the <username>"
echo " -d disable the <roles> for the <username>"
echo " -c display the users with the <roles> assigned"
echo " -C display the users with the <roles> assigned and enabled"
echo " -Q display the users with the <roles> assigned and disabled"
echo " -s show the roles assigned to the user"
echo " -S show the roles assigned and enabled to the user"
echo " -D show the roles assigned and disabled for the user"
echo ""
echo " -f display the roles in a format to be stored in files"
echo " -u user names; default user is [${user_default}]"
echo " -F input text file with usernames one per line at the beginning; comments
start with #"
if [ -z "${role_category}" ]; then
echo " -m authenticate as <userBindDN> to the LDAP server; default is
AccessManagerAdmin"
echo " -M authenticate as [${managerDN}] to the LDAP server"
echo " -p password to bind to the LDAP server"
echo " -l use this ldapserver; default is [${ldapserver}]"
echo " -b use this basedn; default is [${basedn}]"
echo " -v verbose mode"
fi
echo ""
echo "Author: mleahu@CERN"
exit 0;;
G) role_category=${OPTARG}; user_default="";;
L) set_op_value ${OP_SHOW_ROLES};;
v) VERBOSE=${TRUE};;
a) set_op_value ${OP_ASSIGN}; roles=( ${OPTARG} );;
A) set_op_value ${OP_ASSIGN_ENABLE}; roles=( ${OPTARG} );;
r) set_op_value ${OP_REVOKE}; roles=( ${OPTARG} );;
e) set_op_value ${OP_ENABLE}; roles=( ${OPTARG} );;
d) set_op_value ${OP_DISABLE}; roles=( ${OPTARG} );;
c) set_op_value ${OP_SHOW_ROLES_CONTENT}; roles=( ${OPTARG}
);role_state="${ROLE_STATE_ASSIGNED}";;
C) set_op_value ${OP_SHOW_ROLES_CONTENT}; roles=( ${OPTARG}
);role_state="${ROLE_STATE_ENABLED}";;
Q) set_op_value ${OP_SHOW_ROLES_CONTENT}; roles=( ${OPTARG}
);role_state="${ROLE_STATE_DISABLED}";;
s) set_op_value ${OP_SHOW_USER_ROLES};role_state="${ROLE_STATE_ASSIGNED}";;
S) set_op_value ${OP_SHOW_USER_ROLES};role_state="${ROLE_STATE_ENABLED}";;
D) set_op_value ${OP_SHOW_USER_ROLES};role_state="${ROLE_STATE_DISABLED}";;
f) output_for_file="${TRUE}";;
u) users=( ${OPTARG} );;
F) input_file=${OPTARG};;
m) userBindDN="uid=${OPTARG},ou=People";;
M) userBindDN=${managerDN};;
p) passwdBind=${OPTARG};;
l) ldapserver=${OPTARG};;
b) basedn=${OPTARG};;
esac
done
shift $(( ${OPTARG} - 1 ))

if [ "${OPERATION}" = "${OP_HELP}" ]; then
$0 -G "${role_category}" -h
exit 0

```

```

fi

userBindDN="{userBindDN},$basedn"
rolesBaseDN="ou=netgroup,$basedn"

if ( ! which ldapsearch > /dev/null 2>&1 )
then
    echo "ldapsearch tool not found!"
    exit 2
fi

ldapoptions="-h ${ldapserver} -b ${rolesBaseDN} -x"
# sort the output based on cn
ldapoptions="{ldapoptions} -S cn -LLL"
ldapfilter="(objectClass=${LDAP_ROLE_OBJECTCLASS})"
ldapwriteoptions="-W "

# display the roles available in LDAP
if [ "${OPERATION}" = "${OP_SHOW_ROLES}" ]; then

ldapfilter="( ${ldapfilter} (cn=${LDAP_ROLE_PREFIX}${LDAP_ROLE_SEPARATOR}${role_category}*))"
ldapattrs="cn"
ldapscope="SUB"
MSG "ROLES AVAILABLE IN LDAP"
[ldap://${ldapserver}/${rolesBaseDN}?${ldapattrs}?${ldapscope}?${ldapfilter}]
if [ -n "${role_category}" ]; then
    MSG "ROLES FROM CATEGORY [${role_category}]"
fi
ldapsearch ${ldapoptions} "${ldapfilter}" ${ldapattrs} | awk "/^${ldapattrs}:/ { print
substr(\$2,4) }"
exit 0
fi

# read the users from file
if [ -n "${input_file}" ]; then
if [ ! -r "${input_file}" ]; then
    echo "ERROR: Can't read the users from input file [${input_file}]"
else
    users_from_file=( `grep -v "^#" ${input_file} | awk '{print $1}' ` )
    INFO "${#users[@]} users from command line arguments"
    INFO "${#users_from_file[@]} users found in input file"
    users=( ${users[@]} ${users_from_file[@]} )
fi
fi

#transform user names from upper case to lower case
for((i=0;i<${#users[@]};i++))
do
    users[$i]=`echo ${users[$i]} | tr '[:upper:]' '[:lower:]'`
done

# update the role names in the list with the category prefix
if [ -n "${role_category}" -a "${#roles[@]}" -gt "0" ]; then
for ((i=0;i<${#roles[@]}; i++)); do
    roles[$i]="${role_category}:${roles[$i]}"
done
echo "### ROLES TO PROCESS: [${roles[@]}]"
fi

if [ "${OPERATION}" != "${OP_SHOW_ROLES_CONTENT}" ]; then
if [ "${#users[@]}" -eq 0 ]; then
    users=( "${user_default}" )
fi

if [ "${#users[@]}" -eq 0 ]; then
    echo "ERROR: User names must be provided!"
    exit 2
else
    INFO "Check user names against LDAP..."
    #check if users are valid
    valid_users=( )
    for user in ${users[@]}
    do
        if ( ! ldapsearch -x -b "uid=$user,ou=People,$basedn" -h "${ldapserver}"
"uid=$user" uid >/dev/null ); then
            MSG "WARNING: Skip user [user] because was not found in LDAP! " 1>&2
        fi
    done
fi
fi

```

```

        else
            valid_users=( ${valid_users[@]} $user )
        fi
    done
    if [ "${#valid_users[@]}" -gt 0 ]; then
        users=( ${valid_users[@]} )
        if [ "${#users[@]}" -gt 1 ]; then
            MSG "TOTAL: ${#users[@]} users to process: [ ${users[@]} ]"
            if [ "${OPERATION}" -gt "${OP_SHOWS}" ];
            then
                read -p "ARE YOU SURE YOU WANT TO PROCESS THEM (y/n)?" -n 1 answer
                echo
                if [ "${answer}" != "y" ]; then
                    echo "Operation aborted!"
                    exit 2
                fi
            fi
        fi
    else
        MSG "No users to process!"
        exit 0
    fi
fi

if [ "${OPERATION}" -gt "${OP_SHOWS}" ]; then

# get the password to bind without
if [ -z "${passwdBind}" ]
then
    echo -e "Password for ${userBindDN}:"
    read -s passwdBind
    if [ -z "${passwdBind}" ];
    then
        echo "Empty password for ${userBindDN} account!"
        exit 1
    fi
fi
ldapwriteoptions=" -w ${passwdBind} "
fi

if [ "${VERBOSE}" = "${TRUE}" ]; then
    ldapwriteoptions="${ldapwriteoptions} -v"
fi

function if_role_exists_assignable()
{
    local _role=$1
    local _options="${ldapoptions}"
    local
    _filter="( (&${ldapfilter}) (&(cn=${LDAP_ROLE_PREFIX}${LDAP_ROLE_SEPARATOR}${_role}) (${LDAP_ROLE_ATTR_ASSIGNABLE}=${LDAP_ROLE_ASSIGNABLE})) )"
    local -a _attr=( "cn" )
    ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep -q ${_role}
}

function is_user_netgroup_member()
{
    local _user=$1
    local _netgroup=$2
    local _options="${ldapoptions}"
    local _filter="( (&${ldapfilter}) (&(${LDAP_USER_ROLE_ATTR}=`format_user_for_ldap_filter ${_user}`)(cn=${_netgroup})) )"
    local -a _attr=( "${LDAP_USER_ROLE_ATTR}" )
    ldapsearch ${_options} "${_filter}" ${_attr[@]} | grep -q ${_user}
}

# should be called like this: add remove user to netgroup <"add"|"delete"> ${user} ${role}
function add_remove_user_to_netgroup()
{
    local _operation=$1
    local _user=$2
    local _netgroup=$3
    if [ "${_operation}" = "add" ]; then

```

```

    if ( is_user_netgroup_member ${_user} ${_netgroup} ); then
        # already added
        MSGN "ALREADY DONE!"
        return 0
    fi
elif [ "${_operation}" = "delete" ]; then
    if ( ! is_user_netgroup_member ${_user} ${_netgroup} ); then
        # already deleted
        MSGN "ALREADY DONE!"
        return 0
    fi
else
    echo "WRONG OPERATION: ${_operation}"
    return 1
fi

local _ldif="dn: cn=${_netgroup},${rolesBaseDN}
${_operation}: ${LDAP_USER_ROLE_ATTR}
${LDAP_USER_ROLE_ATTR}: `format_user_for_role ${_user}`"
echo "${_ldif}" | ldapmodify -x ${ldapwriteoptions} -D "${userBindDN}" -h ${ldapserver} >
/dev/null
if [ "${PIPESTATUS[1]}" != "0" ]; then
    MSG "FAILED!"
    INFO "LDIF: ${ldap_ldif}"
else
    MSGN "OK!"
fi
}

if [ "${OPERATION}" = "${OP_SHOW_USER_ROLES}" ]; then
    for user in ${users[@]}
    do
        if [ "${#users[@]}" -gt 1 -a "${output_for_file}" = "${FALSE}" ]; then echo "###
${user}"; fi

        #####
        INFO "### DISPLAY USER[${user}]'S ROLES IN STATE [ `echo ${role state} | tr a-z A-Z` ]"
        ###"
        #####
        if [ "${role_state}" = "${ROLE_STATE_DISABLED}" ]; then

filter="((${ldapfilter}) (&(| (cn=${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}*) (cn=${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}*)) (${LDAP_USER_ROLE_ATTR}=`format_user_for_ldap_filter ${user}`)))"
        attr=( "cn" )
        init grep filter ${attr[@]}
        if [ "${output_for_file}" = "${TRUE}" ]; then
            echo -n "$user: "
            ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter} | cut
-d "${LDAP_ROLE_SEPARATOR}" -f 2- | sort | uniq -u | tr '\n' '\t'
            echo
        else
            ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter} | cut
-d "${LDAP_ROLE_SEPARATOR}" -f 2- | sort | uniq -u
        fi
    else
        prefix="${LDAP_ROLE_PREFIX_ASSIGNED}"
        if [ "${role_state}" = "${ROLE_STATE_ENABLED}" ]; then
            prefix="${LDAP_ROLE_PREFIX_ENABLED}"
        fi

filter="((${ldapfilter}) (&(cn=${prefix}${LDAP_ROLE_SEPARATOR}*) (${LDAP_USER_ROLE_ATTR}=`format_user_for_ldap_filter ${user}`)))"
        attr=( "cn" )
        init grep filter ${attr[@]}
        if [ "${output_for_file}" = "${TRUE}" ]; then
            echo -n "$user: "
            ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter} | cut
-d "${LDAP_ROLE_SEPARATOR}" -f 2- | sort | tr '\n' '\t'
            echo
        else
            ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter} | cut
-d "${LDAP_ROLE_SEPARATOR}" -f 2- | sort
        fi
    fi
done

```

```

elif [ "${OPERATION}" = "${OP_SHOW_ROLES_CONTENT}" ]; then
#####
INFO "DISPLAY THE CONTENT OF ROLES [${roles[@]}] IN STATE [ `echo ${role state} | tr a-z A-
Z` ]"
#####
prefix="${LDAP_ROLE_PREFIX_ASSIGNED}"
if [ "${role_state}" = "${ROLE_STATE_ENABLED}" ]; then
    prefix="${LDAP_ROLE_PREFIX_ENABLED}"
fi
attr=( "${LDAP_USER_ROLE_ATTR}" )
init_grep_filter ${attr[@]}
for role in ${roles[@]}
do
    if [ "${#roles[@]}" -gt "1" ]; then
        echo "### ROLE [$role]"
    fi
    filter="( &${ldapfilter} (cn=${prefix}${LDAP_ROLE_SEPARATOR}${role}))"

    # check first if the role exists
    if ( ! ldapsearch ${ldapoptions} "${filter}" cn | grep -q "${role}" ); then
        MSG "ROLE [$role] NOT DEFINED!"
        continue;
    fi

    if [ "${role_state}" = "${ROLE_STATE_DISABLED}" ]; then

filter="( &${ldapfilter} ( | (cn=${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}) (cn=${L
LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role})) )"
        if [ "${output_for_file}" = "${TRUE}" ]; then
            ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter} |
sort | cut -d ',' -f2 | tr -d ")" | uniq -u | while read line
        do
            echo "$line: ${role}"
        done
        else
            ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter} |
sort | cut -d ',' -f2 | tr -d ')' | uniq -u
            if [ "${PIPESTATUS[0]}" -ne 0 ]; then
                MSG "ROLE NOT FOUND!"
            fi
        fi
    else
        if [ "${output_for_file}" = "${TRUE}" ]; then
            ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter} |
sort | cut -d ',' -f2 | tr -d ")" | while read line
        do
            echo "$line: ${role}"
        done
        else
            ldapsearch ${ldapoptions} "${filter}" ${attr[@]} | grep -P ${grepfilter} |
sort | cut -d ',' -f2 | tr -d ')'
            if [ "${PIPESTATUS[0]}" -ne 0 ]; then
                MSG "ROLE NOT FOUND!"
            fi
        fi
    fi
done
elif [ "${OPERATION}" = "${OP_ASSIGN}" ]; then
#####
INFO "ASSIGN ROLES "
#####
for user in ${users[@]}
do
    MSG "Assign roles to user [$user]..."
    for role in ${roles[@]}
    do
        MSGN "...role [$role]..."
        if ( ! if_role_exists_assignable ${role} ); then
            echo "SKIPPED! Role doesn't exist!"
            continue
        fi
        echo -n "assign "
        add_remove_user_to_netgroup "add" "${user}"
"${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}"
        echo
    done
done

```



```

done
INFO "DONE"
elif [ "${OPERATION}" = "${OP_ASSIGN_ENABLE}" ]; then
#####
INFO "ASSIGN AND ENABLE ROLES "
#####
for user in ${users[@]}
do
MSG "Assign and enable roles to user [$user]..."
for role in ${roles[@]}
do
MSGN "...role [$role]..."
if ( ! if_role_exists_assignable ${role} ); then
echo "SKIPPED! Role doesn't exist!"
continue
fi
echo -n "assign "
add_remove_user_to_netgroup "add" "${user}"
"${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}"
echo -n " ..enable "
add_remove_user_to_netgroup "add" "${user}"
"${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role}"
echo
done
done
INFO "DONE"
elif [ "${OPERATION}" = "${OP_REVOKE}" ]; then
#####
INFO "REVOKE ROLES "
#####
for user in ${users[@]}
do
MSG "Revoke roles from user [$user]..."
for role in ${roles[@]}
do
MSGN "...role [$role]..."
if ( ! if_role_exists_assignable ${role} ); then
echo "SKIPPED! Role doesn't exist!"
continue
fi
if ( ! is_user_netgroup_member ${user}
"${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}" ); then
echo "SKIPPED! Role not assigned!"
continue
fi
echo -n "disable "
add_remove_user_to_netgroup "delete" "${user}"
"${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role}"
echo -n " ...revoke "
add_remove_user_to_netgroup "delete" "${user}"
"${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}"
echo
done
done
INFO "DONE"
elif [ "${OPERATION}" = "${OP_ENABLE}" ]; then
#####
INFO "ENABLE ROLES "
#####
for user in ${users[@]}
do
MSG "Enable roles for user [$user]..."
for role in ${roles[@]}
do
MSGN "...role [$role]..."
if ( ! is_user_netgroup_member ${user}
"${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}" ); then
echo "SKIPPED! Role not assigned!"
continue
fi
echo -n "enable "
add_remove_user_to_netgroup "add" "${user}"
"${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role}"
echo
done
done
INFO "DONE"

```

```

elif [ "${OPERATION}" = "${OP_DISABLE}" ]; then
#####
INFO "DISABLE ROLES "
#####
for user in ${users[@]}
do
MSG "Disable roles for user [$user]..."
for role in ${roles[@]}
do
MSGN "...role [$role]..."
if ( ! is_user_netgroup_member ${user}
"${LDAP_ROLE_PREFIX_ASSIGNED}${LDAP_ROLE_SEPARATOR}${role}" ); then
echo "SKIPPED! Role not assigned!"
continue
fi
if ( ! is_user_netgroup_member ${user}
"${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role}" ); then
echo "SKIPPED! Role not enabled!"
continue
fi
echo -n "disable "
add_remove_user_to_netgroup "delete" "${user}"
"${LDAP_ROLE_PREFIX_ENABLED}${LDAP_ROLE_SEPARATOR}${role}"
echo
done
done
INFO "DONE"
fi
exit 0

```

8.6 Login restriction enforcement shell script

```

#!/bin/bash

# Retrieve the AM rules from LDAP to restrict the login access on cluster nodes based on
roles/netgroups.
# Version
# 03/06/2008 [mleahu]: Initial version
# 16/06/2008 [mleahu]: Added support for restrictions with POSIX groups
# 25/06/2008 [mleahu]: Added flag to restart the ssh service
# 05/09/2008 [mleahu]: Touch the crontab file if not exists
# 08/09/2008 [mleahu]: Write the output of cronjob to /var/bwm/amLoginRestriction
# 08/09/2008 [mleahu]: Create the crontab if dir doesn't exist
# 07/10/2008 [mleahu]: Exits if the ldap search fails
# 13/10/2008 [mleahu]: restart the crond service after the crontab update
# 12/05/2009 [lvalsan]: Added support for the uri configuration directive in /etc/ldap.conf

version='${Revision: 1.18 $}'

#get the script full path
pushd `dirname $0`>/dev/null
c_dir=`pwd`
popd >/dev/null

TRUE=1
FALSE=0
VERBOSE=${FALSE}

OP_SHOW_USERS=${FALSE}
OP_ACCESS_CONF_BAK=${FALSE}

OP_CRONJOB_REMOVE=${FALSE}
OP_CRONJOB=0
OP_RESTART_SERVICES=${FALSE}
cron_cmd_flags="-u"
cron_cmd="${c_dir}/`basename $0` ${cron_cmd_flags}"
crontab_file="/var/spool/cron/`whoami`"
crontab_log="/var/bwm/`basename $0`"

function INFO ()
{

```

```

    if [ "$VERBOSE" = "$TRUE" ]; then
        echo "$@"
    fi
}

function MSG()
{
    echo -n ">>> "
    echo "$@"
}

function MSGN()
{
    echo -n ">>> "
    echo -n "$@"
}

machine_name=`hostname -s`
ldapserver=`awk '/^host/ {printf $2}' /etc/ldap.conf`
basedn=`awk '/^base/ {printf $2}' /etc/ldap.conf`
ldapuri=`awk '/^uri/ {printf $2}' /etc/ldap.conf`
#ldapurl="ldap://${ldapserver}/${rolesBaseDN}?${ldapattrs}?${ldapscope}?${ldapfilter}"
ldapurl="ldap://${ldapserver}/${basedn}"

LDAP_RULE_OBJ="sudoRole";
LDAP_RULE_NAME_ATTR_PREFIX="LOGIN-";
LDAP_RULE_HOST_ATTR="sudoHost";
LDAP_RULE_USER_ATTR="sudoUser";

logger_cmd="/usr/bin/logger -p local3.info -t `basename $0`"

access_conf_file="/etc/security/access.conf"
access_conf="#"
# Login access control table.
# Generated automatically by the script '$0' run as `whoami` on the node `hostname` on
# `date`.
#"

access_conf_default="
+ | ALL | ALL
"
access_conf_end="
- | ALL EXCEPT root | ALL"

update_access_conf=$FALSE

# check the command line arguments
while getopts "n:uBsc:Cr1:H:b:vh" option
do
    case $option in
        h) echo "`basename $0` - $version";
           echo "Get the AM specific rules from LDAP to restrict the login on cluster nodes
based on roles/netgroups.";
           echo "Usage: `basename $0` [-n machine_name] [-u] [-B] [-s] [-c minutes] [-C] [-r]
[-l ldapserver] [-H ldapuri] [-b basedn] [-v] [-h]";
           echo "  -n specify the machine where the rule applies. Default is the current
machine."
           echo "  -u update the file [${access_conf_file}]"
           echo "  -B make a backup copy of [${access_conf_file}] before update"
           echo "  -s show the users allowed to login to the machine_name as specified in
LDAP rules"
           echo "  -c set a cronjob in [${crontab_file}] on the current machine to run the
command [${cron_cmd}] every 'minutes' minutes"
           echo "  -C remove the cronjob set with -c if any found. If -c provided in the same
time, then first the current cronjob is removed."
           echo "  -r restart the services with PAM support (e.g., sshd)"
           echo "  -l use this ldapserver; default is [${ldapserver}]"
           echo "  -H use this ldapuri; default is [${ldapuri}]"
           echo "  -b use this basedn; default is [${basedn}]"
           echo "  -v verbose mode"
           echo "  -h this info"
           echo ""
           echo "Author: mleahu@CERN"
           exit 0;;
        n) machine_name=$OPTARG;;
        u) update_access_conf=$TRUE;;
        B) OP_ACCESS_CONF_BAK=$TRUE;;
    esac
done

```

```

s) OP_SHOW_USERS=$TRUE;;
c) OP_CRONJOB=$OPTARG;;
C) OP_CRONJOB_REMOVE=$TRUE;;
r) OP_RESTART_SERVICES=$TRUE;;
l) ldapserver=$OPTARG;;
H) ldapuri=$OPTARG;;
b) basedn=$OPTARG;;
v) VERBOSE=$TRUE;;
*) echo "Unknown option '$option'. Ignored.";;
esac
done
shift $(( $OPTIND - 1 ))

if [ -n "${ldapuri}" ]; then
    connString="-H ${ldapuri}"
else
    connString="-h ${ldapserver}"
fi

declare -a machine_name_filters=( );
# the machine name contains more fields separated by '-' character
function generate_machine_name_filters() {
    local _machine_name="$1"
    local -a _fields=( `echo ${_machine_name} | tr '-' ' '` )
    local _prev_field=""
    local _new_field=""
    machine_name_filters=( )
    for _field in ${_fields[@]}; do
        _new_field="${_prev_field}${_field}"
        _prev_field="${_new_field}-"
        if [ "${_new_field}" != "${_machine_name}" ]; then
            _new_field="${_new_field}-"
        fi
    done
    machine_name_filters=( ${machine_name_filters[@]} "${_new_field}" )
}

declare ldap_search_result="";
# get the users and netgroups allowed to login to the set of machines specified by the
machine_name_filters
function search_ldap_for_machine_name_filters() {
    local _machine_name_filter="";
    local _ldap_opt="${connString} -b ${basedn} -x -LLL -S cn ";
    local _ldap_filter="";
    local -a _ldap_attr=( "${LDAP_RULE_USER_ATTR}" "cn" );

    # prepare the ldap filter from the machine names filter array
    if [ "${#machine_name_filters[@]}" = "0" ]; then
        MSG "No machine name filters!";
        return 1;
    fi
    for((li=0; li<${#machine_name_filters[@]}; li++); do
        _machine_name_filter="${machine_name_filters[$li]}"
        # the last element is the machine name
        if [ "${(li+1)}" -lt "${#machine_name_filters[@]}" ]; then
            _machine_name_filter="${_machine_name_filter}\*"
        fi

        if [ -z "${_ldap_filter}" ]; then
            _ldap_filter="${LDAP_RULE_HOST_ATTR}=${_machine_name_filter}"
        else
            _ldap_filter="| (${LDAP_RULE_HOST_ATTR}=${_machine_name_filter}) (${_ldap_filter})"
        fi
    done

    _ldap_filter="(&(${_ldap_filter}) (&(cn=${LDAP_RULE_NAME_ATTR_PREFIX}*) (objectClass=${LDAP_RULE_OBJ})))"

    ldapurl="ldap://${connString}/${basedn}?${_ldap_attr[@]}?SUB?${_ldap_filter}"

    if [ "$VERBOSE" = "$TRUE" ]; then
        INFO "ldapsearch ${_ldap_opt} ${_ldap_filter} ${_ldap_attr[@]} ${LDAP_RULE_HOST_ATTR}"
        ldapsearch ${_ldap_opt} "${_ldap_filter}" ${_ldap_attr[@]} ${LDAP_RULE_HOST_ATTR}
    fi
}

```

```

ldap search result=`( ldapsearch ${_ldap_opt} "${_ldap_filter}" "${_ldap_attr[@]}" || echo
"LDAP_SEARCH_ERROR[\${?}]" ) 2>&1`
}

declare access_conf_generated="";
# generate the access_conf content
function generate_access_conf() {
    access_conf_generated="";
    access_conf_generated=`echo "${ldap_search_result}" | awk '
        /dn:/ {
            print "";
            print "# SUDO RULE DN: " $2;
        }
        /sudoUser:/ {
            if ( $2 ~ /^\/\+.*\/ ) print "+ | @" substr($2,2) " | ALL";
            else if ( $2 ~ /^\/\%.*\/ ) print "+ | " substr($2,2) " | ALL";
            else print "+ | " $2 " | ALL";
        }
    '`
}

#show the users members of those netgroups
function show_users_allowed() {
    if [ -z "${ldap_search_result}" ]; then
        MSG "No login restriction rules found in LDAP!"
        return
    fi
    echo "${ldap_search_result}" | grep "^sudoUser:" | sort | uniq | while read _sudouser
    _user
    do
        # check if the sudoUser is in fact a netgroup, or unix group, else is an user
        if ( echo "${_user}" | grep -q "^+" ); then
            local _netgroup=`echo ${_user} | tr -d '+'`
            MSG "Netgroup [${_netgroup}]:"
            getent netgroup ${_netgroup} | tr '(\n' | grep "," | tr -d "[:blank:;])" | sort
        elif ( echo "${_user}" | grep -q "^%" ); then
            local _ldap_opt="${connString} -b ${basedn} -x -LLL ";
            local _group=`echo ${_user} | tr -d '%'`
            MSG "Group [${_group}] content from LDAP:"
            ldapsearch ${_ldap_opt} "(&(objectClass=posixGroup)(cn=${_group}))" memberUid | awk
            '/memberUid/ {print $2}' | sort
        else
            MSG "USER:"
            echo ${_user}
        fi
    done
}

declare crontab_minutes=""
function generate_crontab_minutes() {
    local interval=${1}
    #restrict interval to 1...60
    interval=$((interval % 60))
    if [ "${interval}" = "0" ]; then interval=1; fi

    local random=${RANDOM}
    local offset=$((random % interval))
    local max=59

    crontab_minutes="$offset"
    for ((i=$((offset+interval)); i<=${max}; i+=interval))
    do
        crontab_minutes="${crontab_minutes},$i"
    done
}

MSG " ===== LOGIN RESCTRITION CONFIGURATION ====="

if [ "${OP_CRONJOB_REMOVE}" = "$TRUE" ]; then
    MSG "Remove command [${cron_cmd}] from crontab file ..."
    if [ -w "${crontab_file}" ]; then
        if (grep -q "${cron_cmd}" ${crontab_file} ); then
            new_crontab_file=`grep -v ${cron_cmd} ${crontab_file}`
            if [ -z "${new_crontab_file}" ]; then
                # remove the cron for this user
                rm -fv ${crontab_file}
            else

```

```

        echo "${new_crontab_file}" > ${crontab_file}
    fi
    touch `dirname ${crontab_file}`
    MSG "CRONTAB [${crontab_file}]: the command has been REMOVED!"
else
    MSG "CRONTAB [${crontab_file}]: the command has not been found!"
fi
else
    MSG "CRONTAB [${crontab_file}]: not found!"
fi

if [ "${update_access_conf}" = "$FALSE" -a "${OP_CRONJOB}" = "0" ]; then
    # don't do anything more, just exit
    exit 0
fi

fi

#check first if a cronjob has to be set
if [ "${OP_CRONJOB}" -gt 0 ]; then
    MSG "Set a cron job to run every ${OP_CRONJOB} minutes..."

    generate_crontab_minutes ${OP_CRONJOB}

    crontab_log_dir=`dirname ${crontab_log}`
    if [ -n "${crontab_log_dir}" -a ! -d "${crontab_log_dir}" ];
    then
        mkdir -pv "${crontab_log_dir}"
    fi
    crontab_cmd="${crontab_minutes} * * * * ${cron_cmd} > ${crontab_log}";
    echo "CRONJOB [${crontab_cmd}]"

    if [ ! -w "${crontab_file}" ]; then
        echo "Crontab file not writable. Try to touch it ..."
        touch ${crontab_file}
    fi

    if [ -w "${crontab_file}" ]; then

        if ( grep -q "${crontab_cmd}" ${crontab_file} 2>/dev/null ); then
            MSG "CRONTAB [${crontab_file}]: already contains the command!"
        else
            echo "${crontab_cmd}" >> ${crontab_file}
            touch `dirname ${crontab_file}`
            /sbin/service crond restart
            MSG "CRONTAB [${crontab_file}]: updated with the command!"
        fi
    else
        MSG "CRONTAB [${crontab_file}]: not writable!"
    fi

    if [ "${update_access_conf}" = "$FALSE" ]; then
        # don't do anything more, jut exit
        exit 0
    fi

fi

generate_machine_name_filters ${machine_name}
search ldap for machine name filters

#check if the ldap search has failed
if ( echo "${ldap_search_result}" | grep -q "LDAP_SEARCH_ERROR" ); then
    MSG "Update aborted due to ldap search error [${ldapurl}]: [${ldap_search_result}]"
    if [ "${update_access_conf}" = "$TRUE" ]; then
        ${logger_cmd} "Update aborted due to ldap search error [${ldapurl}]:
[${ldap_search_result}]"
    fi
    exit 1
fi

if [ "${OP_SHOW_USERS}" = "$TRUE" ]; then
    # display the users allowed to login to the given machine name
    MSG "Users allowed to login to [${machine_name}]:"
    show_users_allowed
    exit 0;
fi

generate_access_conf

```

```

if [ -n "${access_conf_generated}" ]; then
    MSG "PAM ACCESS configuration generated from LDAP information!"
    access_conf="${access_conf}"
    ${access_conf_generated}
    ${access_conf_end}"
else
    MSG "PAM ACCESS configuration set to default! (nothing found in LDAP for it)"
    access_conf="${access_conf}"
    ${access_conf_default}"
fi

if [ "${update_access_conf}" = "$TRUE" ]; then
    MSGN "Check if update is necessary ..."
    if ( echo "${access_conf}" | diff --brief -q --ignore-matching-lines="^#.*" -
    ${access_conf_file} >/dev/null ); then
        echo "no!"
        exit 0
    fi

    echo "yes!"

    MSG "Update the file [${access_conf_file}]..."
    if [ "`whoami`" != "root" ]; then
        MSG "Only root can do it!"
        exit 0
    fi

    if [ -r "${access_conf_file}" -a "${OP_ACCESS_CONF_BAK}" = "$TRUE" ]; then
        mv -fv ${access_conf_file} ${access_conf_file}.bak-`date +%y%m%d-%H%M`
    fi

    ${logger_cmd} "Generate the file ${access_conf_file} with the new content:"
    echo "${access_conf}" | grep -v "^#" | while read line
    do
        ${logger_cmd} ${line}
    done

    echo "${access_conf}" > ${access_conf_file}

    if [ "${OP_RESTART_SERVICES}" = "$TRUE" ]; then
        MSG "Restart the PAM aware services that need login restriction"
        MSG "[ssh]"
        /sbin/service sshd restart
    fi
else
    echo "${access_conf}"
fi

```

8.7 XACML policies generation by PAP

The preparation of access control policies for the AM server in the XACML format is achieved by running the `amPAP` shell script. Assuming that default values of script parameters correspond to the application setup, then the script can be called without any other parameters.

The application takes as input:

- the `AMRules.txt` file which contains the AM permissions and permissions assignment to roles
- the roles definition and hierarchy in the central LDAP server. Sample roles hierarchy shown in Figure 72.

The output consists in the XML files containing the XACML policies.

The following chapters list the sample input file and the generated policies in XACML format: `pp`, `ppsrules`, `rps`, `pproles`.

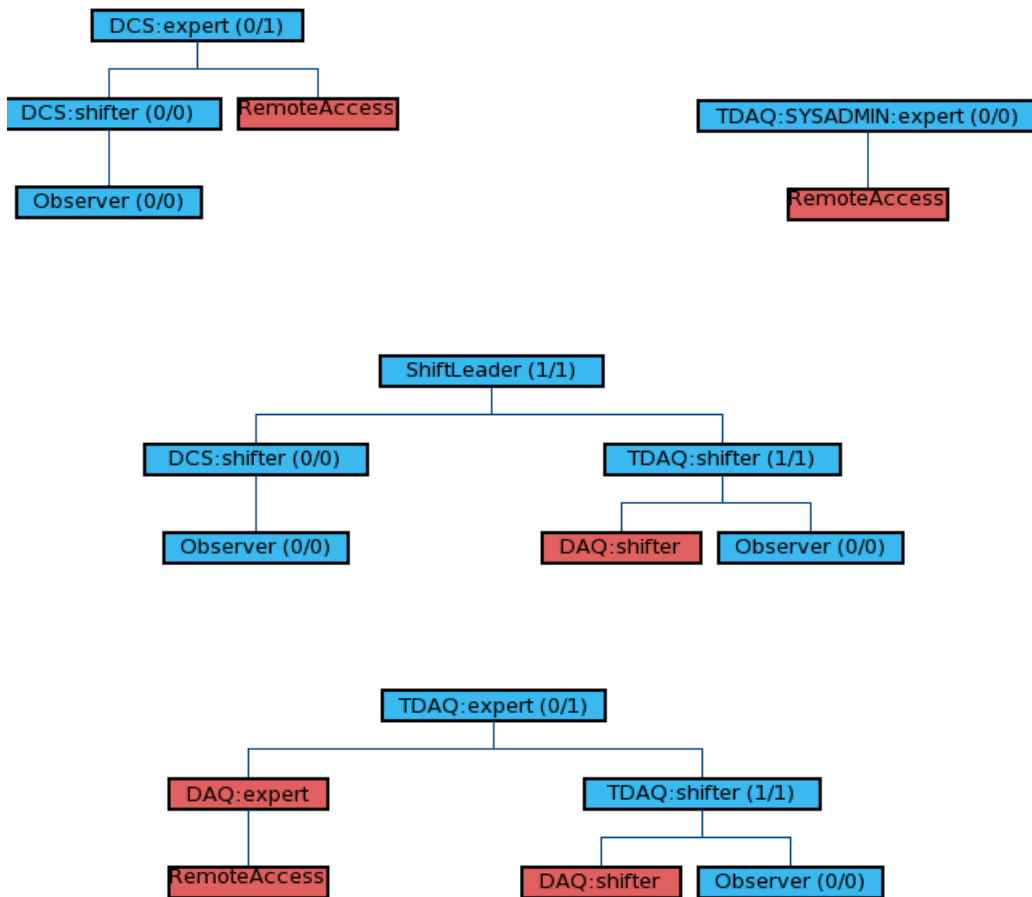


Figure 72 Sample roles hierarchy in LDAP

8.7.1 Input policies

AMRules.txt

```
# Version: $Revision: 53911 $
#
# The lines starting with # are comments
# The rules should be in the following format:
# [Rule=_the_rule_name_] [ResourceCategory=] [ResourceId=] [ResourceType=]
# [<parameter_specific_to_some_resources>] [ActionId=_action_]
# NOTE: a [Decision=decision_value] can be specified with valid values: Permit, Deny. Default
# is "Permit".
# NOTE: regular expression values must have the prefix '{regexp}'

##### Data Base #####
#
# [Rule=_the_rule_name_] [ResourceCategory=DataBase] [ResourceId=_resource_id_]
# [ResourceTypePath= path ] [ActionId= action ]
# resource id can be 'admin', 'directory' or 'file'
# _path_ is the path to the file or directory (not required for admin)
# _action_ can be 'admin', 'create_file', 'update_file', 'delete_file', 'create_subdir',
# 'delete_subdir'
#
[Rule=DAQ:db:daq_dir] [ResourceCategory=DataBase] [ResourceId=directory]
[ResourceTypePath={regexp}^daq/.*) [ActionId=create_subdir]
[Rule=DAQ:db:daq_dir] [ResourceCategory=DataBase] [ResourceId=directory]
[ResourceTypePath={regexp}^daq/.*) [ActionId=delete_subdir]
[Rule=DAQ:db:daq_dir] [ResourceCategory=DataBase] [ResourceId=directory]
[ResourceTypePath={regexp}^daq/.*) [ActionId=create_file]
[Rule=DAQ:db:daq_dir] [ResourceCategory=DataBase] [ResourceId=directory]
[ResourceTypePath={regexp}^daq/.*) [ActionId=delete_file]
#
[Rule=DAQ:db:daq_xml] [ResourceCategory=DataBase] [ResourceId=file]
[ResourceTypePath={regexp}^daq/.*)\.xml] [ActionId=update_file]
#
[Rule=DAQ:db:daq_partitions_xml] [ResourceCategory=DataBase] [ResourceId=file]
[ResourceTypePath={regexp}^daq/partitions/.*)\.xml] [ActionId=update_file]
#
##### Operating System #####
#
# [Rule= the rule name ] [ResourceCategory=os] [ResourceId= resource location ]
# [ResourceType=_application_] [ResourceTypeArguments=_application_arguments_]
# [ActionId=_action_]
#
[Rule=crd:shell] [ResourceCategory=os] [ResourceId=crd] [ResourceType=shell] [ActionId=open]
[Rule=crd:lockscreen] [ResourceCategory=os] [ResourceId=crd] [ResourceType=lockscreen] [ActionId=lock]
[Rule=crd:lockscreen] [ResourceCategory=os] [ResourceId=crd] [ResourceType=lockscreen] [ActionId=unlock]
[Rule=crd:am_tool] [ResourceCategory=os] [ResourceId=crd] [ResourceType=am_tool] [ActionId=role_state_change]
[Rule=remoteaccess] [ResourceCategory=os] [ResourceId=gateway] [ResourceType=login] [ActionId=remote]
#
##### Process Manager #####
#
# [Rule= the rule name ] [ResourceCategory=pmg] [ResourceId= process binary path ]
# [ResourceTypeHostname= process hostname ] [ResourceTypeArguments= process arguments ]
# [ResourceTypeOwnedByRequester=true/false] [ActionId=_action_]
# The _action_ can be: "start", "terminate"
#
[Rule=DAQ:pmg:allow if owned] [ResourceCategory=pmg] [ActionId=start]
[Rule=DAQ:pmg:allow if owned] [ResourceCategory=pmg]
[ResourceTypeOwnedByRequester=true] [ActionId=terminate]
[Rule=DAQ:pmg:allow_all] [ResourceCategory=pmg] [ActionId=start]
[Rule=DAQ:pmg:allow_all] [ResourceCategory=pmg] [ActionId=terminate]
#
##### Resource Manager #####
#
# [Rule=_the_rule_name_] [ResourceCategory=ResourceManager]
# [ResourceTypePartition=_partition_name_] [ActionId=_action_]
# The _action_ can be: "lock", "free"
#
[Rule=DAQ:rm:free_except_initial] [ResourceCategory=ResourceManager] [ActionId=free]
[Rule=DAQ:rm:free_except_initial] [ResourceCategory=ResourceManager]
[ResourceTypePartition=initial] [ActionId=free] [Decision=Deny]
[Rule=DAQ:rm:allow_all] [ResourceCategory=ResourceManager] [ActionId=lock]
```

```
[Rule=DAQ:rm:allow_all] [ResourceCategory=ResourceManager] [ActionId=free]
#
##### Run Control #####
#
# [Rule=_the_rule_name_] [ResourceCategory=RunControl] [ResourceTypeCommand=_command_]
[ResourceTypePartition=_partition_name_] [ActionId=exec_cmd]
#
[Rule=DAQ:rc:publish] [ResourceCategory=RunControl] [ResourceTypeCommand=publish]
[ActionId=exec_cmd]
[Rule=DAQ:rc:publish] [ResourceCategory=RunControl] [ResourceTypeCommand=publish_statistics]
[ActionId=exec_cmd]
[Rule=DAQ:rc:allow_all] [ResourceCategory=RunControl] [ActionId=exec_cmd]
#
##### IGUI #####
#
# [Rule= the rule name ] [ResourceCategory=igui] [ResourceId= view mode ]
[ResourceType=_application_] [ActionId=view]
# The _view_mode can be: "display", "control", "expert"
#
[Rule=DAQ:igui:display] [ResourceCategory=igui] [ResourceId=display] [ActionId=view]
[Rule=DAQ:igui:control] [ResourceCategory=igui] [ResourceId=control] [ActionId=view]
[Rule=DAQ:igui:expert] [ResourceCategory=igui] [ResourceId=expert] [ActionId=view]
#
##### BCM Panel #####
#
# [Rule= the rule name ] [ResourceCategory=BCM] [ResourceId=bcm]
[ActionId=_action_]
# The _action_ can be: "configure", "update"
#
##### RULES ASSIGNMENTS TO ROLES #####
#
# [Role=_the_role_name_] [IncludeRule=_rule_1] [IncludeRule=_rule_2]
#
# The roles below must be defined in the LDAP roles hierarchy
#-----
# ROLE = DAQ:shifter
[Role=DAQ:shifter] [IncludeRule=DAQ:pmg:allow_all] [IncludeRule=DAQ:igui:display]
[IncludeRule=DAQ:igui:control] [IncludeRule=DAQ:igui:expert]
[IncludeRule=DAQ:rc:publish]
[Role=DAQ:shifter] [IncludeRule=DAQ:rm:free_except_initial]
[Role=DAQ:shifter] [IncludeRule=crd:lockscreen] [IncludeRule=crd:shell]
#-----
# ROLE = DAQ:expert
[Role=DAQ:expert] [IncludeRule=DAQ:pmg:allow_all] [IncludeRule=DAQ:rc:allow_all]
[IncludeRule=DAQ:rm:allow_all]
[Role=DAQ:expert] [IncludeRule=crd:shell]
#-----
# ROLE = DCS:shifter
[Role=DCS:shifter] [IncludeRule=crd:shell]
#-----
# ROLE = TDAQ:expert
[Role=TDAQ:expert] [IncludeRule=DAQ:db:daq_dir] [IncludeRule=DAQ:db:daq_xml]
#-----
# ROLE = TDAQ:shifter
[Role=TDAQ:shifter] [IncludeRule=DAQ:db:daq_partitions_xml]
#-----
# ROLE = TDAQ:SYSADMIN:expert
[Role=TDAQ:SYSADMIN:expert] [IncludeRule=crd:shell] [IncludeRule=crd:lockscreen]
#-----
# ROLE = RemoteAccess
[Role=RemoteAccess] [IncludeRule=remoteaccess]
#-----
# ROLE = ShiftLeader
[Role=ShiftLeader] [IncludeRule=crd:am_tool]
```

8.7.2 (PP) Permission Policies for rules

8.7.2.1 crd

pp\crd\am_tool\Rcrd\A.xml

```

<Policy PolicyId="pp:crd:am_tool?Rcrd/A"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule crd:am_tool</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">crd</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="rule?Ram_tool=Arole_state_change" Effect="Permit">
    <Description>Rule for </Description>
    <Target>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">am_tool</AttributeValue>
            <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:application"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">role_state_change</AttributeValue>
            <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>

```

pp\crd\lockscreen\Rcrd\A.xml

```

<Policy PolicyId="pp:crd:lockscreen?Rcrd/A"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule crd:lockscreen</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">crd</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="rule?Rlockscreen=Alock" Effect="Permit">
    <Description>Rule for </Description>
    <Target>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">lockscreen</AttributeValue>
            <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:application"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
  </Rule>
</Policy>

```

```

        <Actions>
          <Action>
            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">lock</AttributeValue>
              <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
    <Rule RuleId="rule?Rlockscreen=Aunlock" Effect="Permit">
      <Description>Rule for </Description>
      <Target>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">lockscreen</AttributeValue>
              <ResourceAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:application"
                DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">unlock</AttributeValue>
              <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>

```

pp\crd\shell\Rcrd\A.xml

```

<Policy PolicyId="pp:crd:shell?Rcrd/A" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
combining-algorithm:deny-overrides">
  <Description>Permission policy for rule crd:shell</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">crd</AttributeValue>
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="rule?Rshell=Aopen" Effect="Permit">
    <Description>Rule for </Description>
    <Target>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string">shell</AttributeValue>
            <ResourceAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:application"
              DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

```

```

        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">open</AttributeValue>
        <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
    </Action>
</Actions>
</Target>
</Rule>
</Policy>

```

8.7.2.2 DAQ

pp\DAQ\db\daq_dir\Rdirectory\A.xml

```

<Policy PolicyId="pp:DAQ:db:daq_dir?Rdirectory/A"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:db:daq_dir</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">directory</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
    <Rule RuleId="rule?Rdaq=Acreate subdir" Effect="Permit">
      <Description>Rule for DB Resource</Description>
      <Target>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">^daq/.*</AttributeValue>
              <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:path"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
              </ResourceMatch>
            </Resource>
          </Resources>
          <Actions>
            <Action>
              <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">create_subdir</AttributeValue>
                <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </ActionMatch>
              </Action>
            </Actions>
          </Target>
        </Rule>
        <Rule RuleId="rule?Rdaq=Adelete subdir" Effect="Permit">
          <Description>Rule for DB Resource</Description>
          <Target>
            <Resources>
              <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
                  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">^daq/.*</AttributeValue>
                  <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:path"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                  </ResourceMatch>
                </Resource>
              </Resources>
              <Actions>
                <Action>

```

```

        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">delete subdir</AttributeValue>
            <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
    </Action>
</Actions>
</Target>
</Rule>
<Rule RuleId="rule?Rdaq=Acreate_file" Effect="Permit">
    <Description>Rule for DB Resource</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
                    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">^daq/.*/</AttributeValue>
                    <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:path"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </ResourceMatch>
            </Resource>
        </Resources>
        <Actions>
            <Action>
                <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">create_file</AttributeValue>
                    <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </ActionMatch>
            </Action>
        </Actions>
    </Target>
</Rule>
<Rule RuleId="rule?Rdaq=Adelete_file" Effect="Permit">
    <Description>Rule for DB Resource</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
                    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">^daq/.*/</AttributeValue>
                    <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:path"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </ResourceMatch>
            </Resource>
        </Resources>
        <Actions>
            <Action>
                <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">delete_file</AttributeValue>
                    <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </ActionMatch>
            </Action>
        </Actions>
    </Target>
</Rule>
</Policy>

```

pp\DAQ\db\daq_partitions_xml\Rfile\A.xml

```

<Policy PolicyId="pp:DAQ:db:daq_partitions_xml?Rfile/A"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
    <Description>Permission policy for rule DAQ:db:daq_partitions_xml</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

```

```

        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">file</AttributeValue>
        <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
    </Resource>
</Resources>
</Target>
<Rule RuleId="rule?Rdaqpartitionsxml=Aupdate file" Effect="Permit">
    <Description>Rule for DB Resource</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
                    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">^daq/partitions/.*\.xml</AttributeValue>
                    <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:path"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Actions>
                <Action>
                    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">update_file</AttributeValue>
                        <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
    </Rule>
</Policy>

```

pp\DAQ\db\daq_xml\Rfile\A.xml

```

<Policy PolicyId="pp:DAQ:db:daq_xml?Rfile/A"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
    <Description>Permission policy for rule DAQ:db:daq_xml</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">file</AttributeValue>
                    <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
        </Target>
    <Rule RuleId="rule?Rdaqxml=Aupdate_file" Effect="Permit">
        <Description>Rule for DB Resource</Description>
        <Target>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
                        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">^daq/.*\.xml</AttributeValue>
                        <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:path"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Actions>
                <Action>
                    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">update_file</AttributeValue>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
    </Rule>
</Policy>

```

```

        <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
    </Action>
</Actions>
</Target>
</Rule>
</Policy>

```

pp\DAQ\igui\control\Rcontrol\Aview.xml

```

<Policy PolicyId="pp:DAQ:igui:control?Rcontrol/Aview"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:igui:control</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">control</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">view</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    <Rule RuleId="rule" Effect="Permit">
      <Description>Rule for IGUI Resource</Description>
      <Target>
      </Target>
    </Rule>
  </Policy>

```

pp\DAQ\igui\display\Rdisplay\Aview.xml

```

<Policy PolicyId="pp:DAQ:igui:display?Rdisplay/Aview"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:igui:display</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">display</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">view</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    <Rule RuleId="rule" Effect="Permit">
      <Description>Rule for IGUI Resource</Description>
      <Target>
      </Target>
    </Rule>
  </Policy>

```



```

</Rule>
</Policy>

```

pp\DAQ\igui\expert\Rexpert\Aview.xml

```

<Policy PolicyId="pp:DAQ:igui:expert?Rexpert/Aview"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:igui:expert</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">expert</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">view</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId="rule" Effect="Permit">
    <Description>Rule for IGUI Resource</Description>
    <Target>
    </Target>
  </Rule>
</Policy>

```

pp\DAQ\pmg\allow_all\R\Astart.xml

```

<Policy PolicyId="pp:DAQ:pmg:allow_all?R/Astart"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:pmg:allow_all</Description>
  <Target>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">start</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId="rule" Effect="Permit">
    <Description>Rule for PMG Resource</Description>
    <Target>
    </Target>
  </Rule>
</Policy>

```

pp\DAQ\pmg\allow_all\R\Aterminate.xml

```

<Policy PolicyId="pp:DAQ:pmg:allow_all?R/Aterminate"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:pmg:allow_all</Description>
  <Target>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">terminate</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId="rule" Effect="Permit">
    <Description>Rule for PMG Resource</Description>
    <Target>
    </Target>
  </Rule>
</Policy>

```

```

    </Actions>
  </Target>
  <Rule RuleId="rule" Effect="Permit">
    <Description>Rule for PMG Resource</Description>
    <Target>
    </Target>
  </Rule>
</Policy>

```

pp\DAQ\rc\allow_all\Rrc\Aexec_cmd.xml

```

<Policy PolicyId="pp:DAQ:rc:allow_all?Rrc/Aexec_cmd"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:rc:allow_all</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">rc</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">exec_cmd</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId="rule" Effect="Permit">
    <Description>Rule for Run Control Resource</Description>
    <Target>
    </Target>
  </Rule>
</Policy>

```

pp\DAQ\rc\publish\Rrc\Aexec_cmd.xml

```

<Policy PolicyId="pp:DAQ:rc:publish?Rrc/Aexec_cmd"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:rc:publish</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">rc</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">exec_cmd</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId="rule?Rpublish=A" Effect="Permit">
    <Description>Rule for Run Control Resource</Description>
    <Target>
      <Resources>
        <Resource>

```

```

        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">publish</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:command"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
</Rule>
<Rule RuleId="rule?Rpublish_statistics=A" Effect="Permit">
  <Description>Rule for Run Control Resource</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">publish_statistics</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:command"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
</Rule>
</Policy>

```

pp\DAQ\rm\allow_all\Rrm\Afree.xml

```

<Policy PolicyId="pp:DAQ:rm:allow_all?Rrm/Afree"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:rm:allow_all</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">rm</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">free</AttributeValue>
          <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId="rule" Effect="Permit">
    <Description>Rule for RM Resource</Description>
    <Target>
      </Target>
  </Rule>
</Policy>

```

pp\DAQ\rm\allow_all\Rrm\Alock.xml

```

<Policy PolicyId="pp:DAQ:rm:allow_all?Rrm/Alock"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule DAQ:rm:allow_all</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">rm</AttributeValue>

```

```

        <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
    </Resource>
</Resources>
<Actions>
    <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">lock</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ActionMatch>
    </Action>
</Actions>
</Target>
<Rule RuleId="rule" Effect="Permit">
    <Description>Rule for RM Resource</Description>
    <Target>
    </Target>
</Rule>
</Policy>

```

pp\DAQ\rm\free_except_initial\Rrm\Afree.xml

```

<Policy PolicyId="pp:DAQ:rm:free_except_initial?Rrm/Afree"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
    <Description>Permission policy for rule DAQ:rm:free_except_initial</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">rm</AttributeValue>
                    <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </ResourceMatch>
            </Resource>
        </Resources>
        <Actions>
            <Action>
                <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">free</AttributeValue>
                    <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </ActionMatch>
            </Action>
        </Actions>
    </Target>
    <Rule RuleId="rule" Effect="Permit">
        <Description>Rule for RM Resource</Description>
        <Target>
        </Target>
    </Rule>
    <Rule RuleId="rule?Rinitial=A" Effect="Deny">
        <Description>Rule for RM Resource</Description>
        <Target>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">initial</AttributeValue>
                        <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:partition"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
        </Target>
    </Rule>
</Policy>

```

8.7.2.3 Remote Access

pp\remoteaccess\Rgateway\A.xml

```
<Policy PolicyId="pp:remoteaccess?Rgateway/A"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Permission policy for rule remoteaccess</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">gateway</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="rule?Rlogin=Aremote" Effect="Permit">
    <Description>Rule for </Description>
    <Target>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">login</AttributeValue>
            <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-type:application"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">remote</AttributeValue>
            <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

8.7.3 (PPSRules) Permissions Policies Sets for rules

8.7.3.1 crd

ppsrules\crd\am_tool\Ros\A.xml

```
<PolicySet PolicySetId="ppsrules:crd:am_tool?Ros/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [crd:am_tool]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">os</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>pp:crd:am_tool?Rcrd/A</PolicyIdReference>
</PolicySet>
```

ppsrules\crd\lockscreen\Ros\A.xml

```
<PolicySet PolicySetId="ppsrules:crd:lockscreen?Ros/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [crd:lockscreen]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">os</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>pp:crd:lockscreen?Rcrd/A</PolicyIdReference>
</PolicySet>
```

ppsrules\crd\shell\Ros\A.xml

```
<PolicySet PolicySetId="ppsrules:crd:shell?Ros/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [crd:shell]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">os</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>pp:crd:shell?Rcrd/A</PolicyIdReference>
</PolicySet>
```

8.7.3.2 DAQ

DAQ\db\daq_dir\RDataBase\A.xml

```
<PolicySet PolicySetId="ppsrules:DAQ:db:daq_dir?RDataBase/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [DAQ:db:daq_dir]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">DataBase</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>pp:DAQ:db:daq_dir?Rdirectory/A</PolicyIdReference>
</PolicySet>
```

ppsrules\DAQ\db\daq_partitions_xml\RDataBase\A.xml

```
<PolicySet PolicySetId="ppsrules:DAQ:db:daq_partitions_xml?RDataBase/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [DAQ:db:daq_partitions_xml]</Description>
  <Target>
    <Resources>
      <Resource>
```

```

        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">DataBase</AttributeValue>
            <ResourceAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
                DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
            </ResourceMatch>
        </Resource>
    </Resources>
</Target>
<PolicyIdReference>pp:DAQ:db:daq_partitions_xml?Rfile/A</PolicyIdReference>
</PolicySet>

```

ppsrules\DAQ\db\daq_xml\RDataBase\A.xml

```

<PolicySet PolicySetId="ppsrules:DAQ:db:daq_xml?RDataBase/A"
    PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
    applicable">
    <Description>Permission Policy Set for rule [DAQ:db:daq_xml]</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
                        DataType="http://www.w3.org/2001/XMLSchema#string">DataBase</AttributeValue>
                    <ResourceAttributeDesignator
                        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
                        DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
        </Target>
        <PolicyIdReference>pp:DAQ:db:daq_xml?Rfile/A</PolicyIdReference>
    </PolicySet>

```

ppsrules\DAQ\igui\control\Rigui\A.xml

```

<PolicySet PolicySetId="ppsrules:DAQ:igui:control?Rigui/A"
    PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
    applicable">
    <Description>Permission Policy Set for rule [DAQ:igui:control]</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
                        DataType="http://www.w3.org/2001/XMLSchema#string">igui</AttributeValue>
                    <ResourceAttributeDesignator
                        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
                        DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
        </Target>
        <PolicyIdReference>pp:DAQ:igui:control?Rcontrol/Aview</PolicyIdReference>
    </PolicySet>

```

ppsrules\DAQ\igui\display\Rigui\A.xml

```

<PolicySet PolicySetId="ppsrules:DAQ:igui:display?Rigui/A"
    PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
    applicable">
    <Description>Permission Policy Set for rule [DAQ:igui:display]</Description>
    <Target>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
                        DataType="http://www.w3.org/2001/XMLSchema#string">igui</AttributeValue>
                    <ResourceAttributeDesignator
                        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
                        DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
        </Target>
        <PolicyIdReference>pp:DAQ:igui:display?Rdisplay/Aview</PolicyIdReference>
    </PolicySet>

```

```
</PolicySet>
```

ppsrules\DAQ\igui\expert\Rigui\A.xml

```
<PolicySet PolicySetId="ppsrules:DAQ:igui:expert?Rigui/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [DAQ:igui:expert]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">igui</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>pp:DAQ:igui:expert?Rexpert/Aview</PolicyIdReference>
</PolicySet>
```

ppsrules\DAQ\pmg\allow_all\Rpmg\A.xml

```
<PolicySet PolicySetId="ppsrules:DAQ:pmg:allow_all?Rpmg/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [DAQ:pmg:allow_all]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">pmg</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>pp:DAQ:pmg:allow_all?R/Astart</PolicyIdReference>
  <PolicyIdReference>pp:DAQ:pmg:allow_all?R/Aterminate</PolicyIdReference>
</PolicySet>
```

ppsrules\DAQ\rc\allow_all\RRRunControl\A.xml

```
<PolicySet PolicySetId="ppsrules:DAQ:rc:allow_all?RRRunControl/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [DAQ:rc:allow_all]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">RunControl</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>pp:DAQ:rc:allow_all?Rrc/Aexec_cmd</PolicyIdReference>
</PolicySet>
```

ppsrules\DAQ\rc\publish\RRRunControl\A.xml

```
<PolicySet PolicySetId="ppsrules:DAQ:rc:publish?RRRunControl/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [DAQ:rc:publish]</Description>
  <Target>
    <Resources>
```



```

    <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">RunControl</AttributeValue>
        <ResourceAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
          DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>pp:DAQ:rc:publish?Rrc/Aexec_cmd</PolicyIdReference>
</PolicySet>

```

ppsrules\DAQ\rm\allow_all\RResourceManager\A.xml

```

<PolicySet PolicySetId="ppsrules:DAQ:rm:allow_all?RResourceManager/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [DAQ:rm:allow_all]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">ResourceManager</AttributeValue>
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
    <PolicyIdReference>pp:DAQ:rm:allow_all?Rrm/Alock</PolicyIdReference>
    <PolicyIdReference>pp:DAQ:rm:allow_all?Rrm/Afree</PolicyIdReference>
  </PolicySet>

```

ppsrules\DAQ\rm\free_except_initial\RResourceManager\A.xml

```

<PolicySet PolicySetId="ppsrules:DAQ:rm:free_except_initial?RResourceManager/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [DAQ:rm:free_except_initial]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">ResourceManager</AttributeValue>
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
    <PolicyIdReference>pp:DAQ:rm:free_except_initial?Rrm/Afree</PolicyIdReference>
  </PolicySet>

```

8.7.3.3 Remote Access

ppsrules\remoteaccess\Ros\A.xml

```

<PolicySet PolicySetId="ppsrules:remoteaccess?Ros/A"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Permission Policy Set for rule [remoteaccess]</Description>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">os</AttributeValue>
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-category"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
  </PolicySet>

```

```

    </Resource>
  </Resources>
</Target>
<PolicyIdReference>pp:remoteaccess?Rgateway/A</PolicyIdReference>
</PolicySet>

```

8.7.4 (RPS) Role Policies Sets

rps\ATLAS.xml

```

<PolicySet PolicySetId="rps:ATLAS" PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:first-applicable">
  <Description>The root policy for all the roles</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">TDAQ*</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
            DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">DCS*</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
            DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#anyURI">ShiftLeader</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
            DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#anyURI">Observer</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
            DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <PolicySetIdReference>rps:IntermediaryTopRole:DCS</PolicySetIdReference>
  <PolicySetIdReference>rps:RoleAssignable?Observer</PolicySetIdReference>
  <PolicySetIdReference>rps:IntermediaryTopRole:TDAQ</PolicySetIdReference>
  <PolicySetIdReference>rps:RoleAssignable?ShiftLeader</PolicySetIdReference>
</PolicySet>

```

rps\IntermediaryTopRole\DCS.xml

```

<PolicySet PolicySetId="rps:IntermediaryTopRole:DCS"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Intermediate RPS for role prefix [DCS]</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#anyURI">DCS:expert</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
            DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#anyURI">DCS:shifter</AttributeValue>

```

```

        <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
    </SubjectMatch>
</Subject>
</Subjects>
</Target>
<PolicySetIdReference>rps:RoleAssignable?DCS:shifter</PolicySetIdReference>
<PolicySetIdReference>rps:RoleAssignable?DCS:expert</PolicySetIdReference>
</PolicySet>

```

rps\IntermediaryTopRole\TDAQ.xml

```

<PolicySet PolicySetId="rps:IntermediaryTopRole:TDAQ"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
    <Description>Intermediate RPS for role prefix [TDAQ]</Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match">
                    <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">TDAQ:SYSADMIN*</AttributeValue>
                <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
            </SubjectMatch>
        </Subject>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
                    <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#anyURI">TDAQ:shifter</AttributeValue>
                <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
            </SubjectMatch>
        </Subject>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
                    <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#anyURI">TDAQ:expert</AttributeValue>
                <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
            </SubjectMatch>
        </Subject>
        </Subjects>
    </Target>
    <PolicySetIdReference>rps:RoleAssignable?TDAQ:expert</PolicySetIdReference>
    <PolicySetIdReference>rps:IntermediaryTopRole:TDAQ:SYSADMIN</PolicySetIdReference>
    <PolicySetIdReference>rps:RoleAssignable?TDAQ:shifter</PolicySetIdReference>
</PolicySet>

```

rps\IntermediaryTopRole\TDAQ\SYSADMIN.xml

```

<PolicySet PolicySetId="rps:IntermediaryTopRole:TDAQ:SYSADMIN"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
    <Description>Intermediate RPS for role prefix [TDAQ:SYSADMIN]</Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
                    <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#anyURI">TDAQ:SYSADMIN:expert</AttributeValue>
                <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
            </SubjectMatch>
        </Subject>
        </Subjects>
    </Target>
    <PolicySetIdReference>rps:RoleAssignable?TDAQ:SYSADMIN:expert</PolicySetIdReference>
</PolicySet>

```

rps\RoleAssignable\DCS\expert.xml

```

<PolicySet PolicySetId="rps:RoleAssignable?DCS:expert"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
    <Description>Role Policy Set for role DCS:expert</Description>
    <Target>

```

```

<Subjects>
  <Subject>
    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">DCS:expert</AttributeValue>
      <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
    </SubjectMatch>
  </Subject>
</Subjects>
</Target>
<PolicySetIdReference>ppsroles:DCS:expert</PolicySetIdReference>
</PolicySet>

```

rps\RoleAssignable\DCS\shifter.xml

```

<PolicySet PolicySetId="rps:RoleAssignable?DCS:shifter"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Role Policy Set for role DCS:shifter</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">DCS:shifter</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <PolicySetIdReference>ppsroles:DCS:shifter</PolicySetIdReference>
</PolicySet>

```

rps\RoleAssignable\TDAQ\SYSADMIN\expert.xml

```

<PolicySet PolicySetId="rps:RoleAssignable?TDAQ:SYSADMIN:expert"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Role Policy Set for role TDAQ:SYSADMIN:expert</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">TDAQ:SYSADMIN:expert</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <PolicySetIdReference>ppsroles:TDAQ:SYSADMIN:expert</PolicySetIdReference>
</PolicySet>

```

rps\RoleAssignable\TDAQ\expert.xml

```

<PolicySet PolicySetId="rps:RoleAssignable?TDAQ:expert"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Role Policy Set for role TDAQ:expert</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">TDAQ:expert</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <PolicySetIdReference>ppsroles:TDAQ:expert</PolicySetIdReference>
</PolicySet>

```

rps\RoleAssignable\TDAQ\shifter.xml

```
<PolicySet PolicySetId="rps:RoleAssignable?TDAQ:shifter"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Role Policy Set for role TDAQ:shifter</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">TDAQ:shifter</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <PolicySetIdReference>ppsroles:TDAQ:shifter</PolicySetIdReference>
</PolicySet>
```

rps\RoleAssignable\Observer.xml

```
<PolicySet PolicySetId="rps:RoleAssignable?Observer"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Role Policy Set for role Observer</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">Observer</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <PolicySetIdReference>ppsroles:Observer</PolicySetIdReference>
</PolicySet>
```

rps\RoleAssignable\ShiftLeader.xml

```
<PolicySet PolicySetId="rps:RoleAssignable?ShiftLeader"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>Role Policy Set for role ShiftLeader</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">ShiftLeader</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <PolicySetIdReference>ppsroles:ShiftLeader</PolicySetIdReference>
</PolicySet>
```

8.7.5 (PPSroles) Permissions Policies Sets for roles

ppsroles\DAQ\expert.xml

```
<PolicySet PolicySetId="ppsroles:DAQ:expert"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [DAQ:expert]</Description>
  <Target/>
  <PolicySetIdReference>ppsrules:DAQ:rm:allow all?RResourceManager/A</PolicySetIdReference>
  <PolicySetIdReference>ppsroles:RemoteAccess</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:DAQ:pmg:allow all?Rpmg/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:crd:shell?Ros/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:DAQ:rc:allow_all?RRunControl/A</PolicySetIdReference>
```

```
</PolicySet>
```

ppsroles\DAQ\shifter.xml

```
<PolicySet PolicySetId="ppsroles:DAQ:shifter"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [DAQ:shifter]</Description>
<Target/>
  <PolicySetIdReference>ppsrules:DAQ:igui:expert?Rigui/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:crd:lockscreen?Ros/A</PolicySetIdReference>

<PolicySetIdReference>ppsrules:DAQ:rm:free_except_initial?RResourceManager/A</PolicySetIdRefer-
ence>
  <PolicySetIdReference>ppsrules:DAQ:pmg:allow_all?Rpmg/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:crd:shell?Ros/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:DAQ:igui:display?Rigui/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:DAQ:rc:publish?RRunControl/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:DAQ:igui:control?Rigui/A</PolicySetIdReference>
</PolicySet>
```

ppsroles\DCS\expert.xml

```
<PolicySet PolicySetId="ppsroles:DCS:expert"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [DCS:expert]</Description>
<Target/>
  <PolicySetIdReference>ppsroles:RemoteAccess</PolicySetIdReference>
  <PolicySetIdReference>ppsroles:DCS:shifter</PolicySetIdReference>
</PolicySet>
```

ppsroles\DCS\shifter.xml

```
<PolicySet PolicySetId="ppsroles:DCS:shifter"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [DCS:shifter]</Description>
<Target/>
  <PolicySetIdReference>ppsroles:Observer</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:crd:shell?Ros/A</PolicySetIdReference>
</PolicySet>
```

ppsroles\TDAQ\SYSADMIN\expert.xml

```
<PolicySet PolicySetId="ppsroles:TDAQ:SYSADMIN:expert"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [TDAQ:SYSADMIN:expert]</Description>
<Target/>
  <PolicySetIdReference>ppsroles:RemoteAccess</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:crd:lockscreen?Ros/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:crd:shell?Ros/A</PolicySetIdReference>
</PolicySet>
```

ppsroles\TDAQ\expert.xml

```
<PolicySet PolicySetId="ppsroles:TDAQ:expert"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [TDAQ:expert]</Description>
<Target/>
  <PolicySetIdReference>ppsroles:TDAQ:shifter</PolicySetIdReference>
  <PolicySetIdReference>ppsroles:DAQ:expert</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:DAQ:db:daq_xml?RDataBase/A</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:DAQ:db:daq_dir?RDataBase/A</PolicySetIdReference>
</PolicySet>
```

ppsroles\TDAQ\shifter.xml

```
<PolicySet PolicySetId="ppsroles:TDAQ:shifter"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [TDAQ:shifter]</Description>
<Target/>
  <PolicySetIdReference>ppsroles:Observer</PolicySetIdReference>
  <PolicySetIdReference>ppsrules:DAQ:db:daq_partitions_xml?RDataBase/A</PolicySetIdReference>
  <PolicySetIdReference>ppsroles:DAQ:shifter</PolicySetIdReference>
</PolicySet>
```

```
</PolicySet>
```

ppсроles\Observer.xml

```
<PolicySet PolicySetId="ppсроles:Observer"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [Observer]</Description>
<Target/>
</PolicySet>
```

ppсроles\RemoteAccess.xml

```
<PolicySet PolicySetId="ppсроles:RemoteAccess"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [RemoteAccess]</Description>
<Target/>
  <PolicySetIdReference>ppсроles:remoteaccess?Ros/A</PolicySetIdReference>
</PolicySet>
```

ppсроles\ShiftLeader.xml

```
<PolicySet PolicySetId="ppсроles:ShiftLeader"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
applicable">
  <Description>The main Permission Policy Set for role [ShiftLeader]</Description>
<Target/>
  <PolicySetIdReference>ppсроles:TDAQ:shifter</PolicySetIdReference>
  <PolicySetIdReference>ppсроles:DCS:shifter</PolicySetIdReference>
  <PolicySetIdReference>ppсроles:crd:am_tool?Ros/A</PolicySetIdReference>
</PolicySet>
```

8.8 Server statistics log sample

```
[11/11 22:39:21][am.server.statistics.StatisticsCollector:<init>] FINE: Sampling clock object
initialized with sampling time interval 1000 ms
[11/11 22:39:21][am.server.statistics.StatisticsCollector:registerReporter] INFO: Register
new req[new requests to be processed]:0.0 new req/s
[11/11 22:39:21][am.server.statistics.StatisticsCollector:registerReporter] INFO: Register
req_proc_ok[requests processed successfully]:0.0 req_proc_ok/s
[11/11 22:39:21][am.server.statistics.StatisticsCollector:registerReporter] INFO: Register
req_proc_err[requests processed with errors]:0.0 req_proc_err/s
[11/11 22:39:21][am.server.statistics.StatisticsCollector:registerReporter] INFO: Register
Load counter: current load [0 srv_load] last peak [0srv_load_peak]
[11/11 22:39:21][am.server.statistics.StatisticsCollector:registerReporter] INFO: Register
req_when_busy[requests received when the server is busy]:0.0 req_when_busy/s
[11/11 22:39:21][am.server.statistics.StatisticsCollector:registerReporter] INFO: Register
busy resp[busy reponses sent to clients]:0.0busy resp/s
[11/11 22:39:21][am.server.statistics.StatisticsCollector:registerReporter] INFO: Register
busy_resp_err[errors while sending busy responses]:0.0 busy_resp_err/s
[11/11 22:39:22][am.server.statistics.StatisticsCollector:registerReporter] INFO: Register JVM
monitoring:510391504 JVM_free_mem 518979584 JVM_total_mem 518979584 JVM_max_mem
[11/11 22:40:24][am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]: 0.0
new req/s 0.0 req_proc_ok/s 0.0 req_proc_err/s 0 srv_load_peak 0.0
req_when_busy/s 0.0 busy_resp/s 0.0 busy_resp_err/s 501803152
JVM_free_mem 518979584 JVM_total_mem 518979584 JVM_max_mem
[11/11 22:40:25][am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]: 1.0
new req/s 0.0 req_proc_ok/s 0.0 req_proc_err/s 1 srv_load_peak 0.0
req_when_busy/s 0.0 busy_resp/s 0.0 busy_resp_err/s 493215152
JVM_free_mem 518979584 JVM_total_mem 518979584 JVM_max_mem
[11/11 22:40:26][am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]: 3.0
new req/s 3.0 req_proc_ok/s 0.0 req_proc_err/s 1 srv_load_peak 0.0
req_when_busy/s 0.0 busy_resp/s 0.0 busy_resp_err/s 476032080
JVM_free_mem 518979584 JVM_total_mem 518979584 JVM_max_mem
[11/11 22:40:27][am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]: 11.0
new req/s 11.0 req_proc_ok/s 0.0 req_proc_err/s 1 srv_load_peak 0.0
req_when_busy/s 0.0 busy_resp/s 0.0 busy_resp_err/s 467444096
JVM_free_mem 518979584 JVM_total_mem 518979584 JVM_max_mem
[11/11 22:40:28][am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]: 12.0
new req/s 12.0 req_proc_ok/s 0.0 req_proc_err/s 1 srv_load_peak 0.0
req_when_busy/s 0.0 busy_resp/s 0.0 busy_resp_err/s 455974816
JVM_free_mem 518979584 JVM_total_mem 518979584 JVM_max_mem
[11/11 22:40:29][am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]: 13.0
new req/s 13.0 req_proc_ok/s 0.0 req_proc_err/s 1 srv_load_peak 0.0
```

	req_when_busy/s	0.0	busy_resp/s	0.0	busy_resp_err/s	447386440
	JVM_free_mem	518979584	JVM_total_mem	518979584	JVM_max_mem	
[11/11 22:40:30]	[am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]:					12.0
	new_req/s	12.0	req_proc_ok/s	0.0	req_proc_err/s	1
	req_when_busy/s	0.0	busy_resp/s	0.0	busy_resp_err/s	435935248
	JVM_free_mem	518979584	JVM_total_mem	518979584	JVM_max_mem	
[11/11 22:40:31]	[am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]:					12.0
	new_req/s	13.0	req_proc_ok/s	0.0	req_proc_err/s	1
	req_when_busy/s	0.0	busy_resp/s	0.0	busy_resp_err/s	427346720
	JVM_free_mem	518979584	JVM_total_mem	518979584	JVM_max_mem	
[11/11 22:40:33]	[am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]:					11.0
	new_req/s	10.0	req_proc_ok/s	0.0	req_proc_err/s	1
	req_when_busy/s	0.0	busy_resp/s	0.0	busy_resp_err/s	418758384
	JVM_free_mem	518979584	JVM_total_mem	518979584	JVM_max_mem	
[11/11 22:40:35]	[am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]:					9.0
	new_req/s	9.0	req_proc_ok/s	0.0	req_proc_err/s	1
	req_when_busy/s	0.0	busy_resp/s	0.0	busy_resp_err/s	413032920
	JVM_free_mem	518979584	JVM_total_mem	518979584	JVM_max_mem	
[11/11 22:40:36]	[am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]:					10.0
	new_req/s	10.0	req_proc_ok/s	0.0	req_proc_err/s	1
	req_when_busy/s	0.0	busy_resp/s	0.0	busy_resp_err/s	404436720
	JVM_free_mem	518979584	JVM_total_mem	518979584	JVM_max_mem	
[11/11 22:40:37]	[am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]:					6.0
	new_req/s	7.0	req_proc_ok/s	0.0	req_proc_err/s	1
	req_when_busy/s	0.0	busy_resp/s	0.0	busy_resp_err/s	398710224
	JVM_free_mem	518979584	JVM_total_mem	518979584	JVM_max_mem	
[11/11 22:40:37]	[am.server.statistics.StatisticsCollector:update] INFO: Reporters[8]:					0.0
	new_req/s	0.0	req_proc_ok/s	0.0	req_proc_err/s	0
	req_when_busy/s	0.0	busy_resp/s	0.0	busy_resp_err/s	398710224
	JVM_free_mem	518979584	JVM_total_mem	518979584	JVM_max_mem	

Bibliography

- [1] CERN, "European Organization for Nuclear Research," CERN, [Online]. Available: <http://www.cern.ch>.
- [2] CERN, "ATLAS TDAQ SysAdmin Team," [Online]. Available: <http://atlas-tdaq-sysadmin.web.cern.ch>.
- [3] CERN, "LHC First Beam," CERN, 10 September 2008. [Online]. Available: <http://lhc-first-beam.web.cern.ch>.
- [4] CERN, "The LHC is back," CERN, 20 November 2009. [Online]. Available: <http://press.web.cern.ch/press/PressReleases/Releases2009/PR16.09E.html>.
- [5] CERN, "The ATLAS Experiment," CERN, [Online]. Available: <http://atlas.ch>.
- [6] ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider," *Journal of Instrumentation*, vol. 3, no. 08, August 2008.
- [7] ATLAS Collaboration, "ATLAS : technical proposal for a general-purpose pp experiment at the Large Hadron Collider at CERN," CERN, Geneva, 1995.
- [8] CERN, "Latest Results from ATLAS Higgs Search," 2012. [Online]. Available: <http://www.atlas.ch/news/2012/latest-results-from-higgs-search.html>.
- [9] M. A. e. al, "Integration of the trigger and data acquisition systems in ATLAS," *Journal of Physics: Conference Series*, vol. 119, no. 2, 2008.
- [10] ATLAS Collaboration, "ATLAS high-level trigger, data-acquisition and controls : Technical Design Report," CERN, Geneva, 2003.
- [11] G. L. Miotto, I. Aleksandrov, A. Amorim, G. Avolio, E. Badescu, M. Caprini, A. Corso-Radu, G. L. Darlea, A. Dos Anjos, I. Fedorko, A. Kazarov, S. Kolos, V. Kotov, A. J. Lankford, M. Leahu, L. Mapelli, R. M. Garcia and Y. Ryabov, "Configuration and control of the ATLAS trigger and data acquisition," *Nuclear Instruments and Methods in Physics Research Section A*, vol. 623, no. 1, pp. 549-551, 11 2010.
- [12] M. C. Leahu, M. Dobson and G. Avolio, "Access Control Design and Implementations in the ATLAS Experiment," *IEEE Transactions on Nuclear Science*, vol. 55, no. 1, pp. 386 - 391, 8 February 2008.
- [13] OASIS, "OASIS XACML Standard," [Online]. Available: <http://www.oasis-open.org/specs/index.php#xacmlv2.0>.
- [14] T. I. E. T. F. (. -. N. W. Group, "Lightweight Directory Access Protocol (LDAP): The Protocol," June 2006. [Online]. Available: <http://tools.ietf.org/html/rfc4511>.
- [15] NSTISSC, National Information Systems Security (INFOSEC) Glossary, September,2000.
- [16] J. Saltzer and M. Schroeder, "The protection of information in computer systems," September 1975.
- [17] D. F. Ferraiolo, D. R. Kuhn and R. Chandramouli, "Role-Based Access Control," vol. Computer

Security Series, 2003.

- [18] DoD, "Department of Defense Trusted Computer System Evaluation Criteria," 26 December 1985. [Online]. Available: <http://www.fas.org/irp/nsa/rainbow/std001.htm>.
- [19] R. Sandhu, D. Ferraiolo and R. K. D, "The NIST Model for Role Based Access Control: Towards a Unified Standard," Berlin, 2000.
- [20] CERN, "Information Technology Department - Organization," 2012. [Online]. Available: <http://information-technology.web.cern.ch/about/organisation>.
- [21] J. Sloper, M. Leahu, M. Dobson and G. Lehmann, "Access management in the ATLAS TDAQ," *IEEE Transactions on Nuclear Science*, vol. 53, no. 3, pp. 986 - 989, 26 June 2006.
- [22] M. Dobson, M. Ciobotaru, E. Ertorer, H. Garitaonandia, L. Leahu, M. Leahu, I. M. Malciu, E. Panikashvili, A. Topurov and G. Ünel, "The Architecture and Administration of the ATLAS Online Computing System," in *15th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2006)*, Mumbai, 2006.
- [23] CERN, "Scientific Linux CERN 5," [Online]. Available: <http://linux.web.cern.ch/linux/scientific5/>.
- [24] "SCADA System (Supervisory Control & Data Acquisition)," [Online]. Available: <http://www.pvss.com>.
- [25] CERN, "JCOP Framework Access Control component," [Online]. Available: <http://j2eeps.cern.ch/wikis/display/EN/JCOP+Framework+Access+Control>.
- [26] O. Holme, M. Gonzalez-Berges, P. Golonka and S. Schmeling, "The JCOP Framework," in *10th International Conference on Accelerator and Large Experimental Physics Control Systems*, Geneva, 2005.
- [27] CERN, "Training on JCOP Framework - Access Control Component," October 2006. [Online]. Available: <https://edms.cern.ch/file/1056103/1/advancedCourse-AccessControl.pdf>.
- [28] "Linux PAM," [Online]. Available: <http://www.linux-pam.org/>.
- [29] "OpenSSH," [Online]. Available: <http://www.openssh.org/>.
- [30] "Linux Sudo," [Online]. Available: <http://www.sudo.ws/>.
- [31] KDE TechBase, "KDE System Administration/Kiosk/Introduction," 2012. [Online]. Available: http://techbase.kde.org/KDE_System_Administration/Kiosk/Introduction.
- [32] OASIS, "Organization for the Advancement of Structured Information Standards," [Online]. Available: <https://www.oasis-open.org/org>.
- [33] World Wide Web Consortium (W3C), "Extensible Markup Language (XML)," [Online]. Available: <http://www.w3.org/XML/>.
- [34] BeginLinux.com, "Basics of LDAP," [Online]. Available: http://beginlinux.com/server_training/server-managment-topics/1015-basics-of-ldap.
- [35] "OpenLDAP," [Online]. Available: <http://www.openldap.org/>.
- [36] "php LDAP Admin," [Online]. Available: <http://phpldapadmin.sourceforge.net>.
- [37] ORACLE, "System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP),"

- [Online]. Available: <http://docs.oracle.com/cd/E19082-01/819-3194/anis2-14244/index.html>.
- [38] OpenLDAP, "Schema specification," [Online]. Available: <http://www.openldap.org/doc/admin22/schema.html>.
- [39] ORACLE, "NIS Extension Guide: NIS Information in the LDAP Directory," [Online]. Available: <http://docs.oracle.com/cd/E19513-01/806-4251-10/mapping.htm>.
- [40] "Sudo - OpenLDAP schema," [Online]. Available: <http://www.sudo.ws/repos/sudo/file/dacfad7e7a95/doc/schema.OpenLDAP>. [Accessed September 2012].
- [41] "GNU Bash Reference Manual," [Online]. Available: <http://www.gnu.org/software/bash/manual/bashref.html>.
- [42] "PHP," [Online]. Available: <http://www.php.net/>.
- [43] "PHP Tree Graph Ext," [Online]. Available: <http://freecode.com/projects/phptreegraphext>.
- [44] CERN, "IT Department - Public Interactive Logon Service," [Online]. Available: <http://plus.web.cern.ch/plus/>.
- [45] A. Collaboration, "Development of the ATLAS control room," 2006. [Online]. Available: <http://www.atlas.ch/news/2006/control-room.html>.
- [46] "K Desktop Environment," [Online]. Available: <http://www.kde.org/>.
- [47] "KDE Kiosk," [Online]. Available: http://techbase.kde.org/KDE_System_Administration/Kiosk/Introduction.
- [48] "Name Service Switch - The GNU C Library," [Online]. Available: http://www.gnu.org/software/libc/manual/html_node/Name-Service-Switch.html.
- [49] "PAM Access," [Online]. Available: http://linux.die.net/man/8/pam_access.
- [50] "cron," [Online]. Available: <http://linux.die.net/man/8/cron>.
- [51] J. E. Sloper and I. Soloviev, "ATLAS TDAQ Online Software - Access Manager Requirements," CERN, 26 August 2004. [Online]. Available: https://edms.cern.ch/file/484984/2/requirements_draft_v2.pdf.
- [52] J. E. Sloper, "ATLAS TDAQ - Access Manager Architectural Analysis and Design," CERN, 31 January 2005. [Online]. Available: https://edms.cern.ch/file/558196/1.0/AM_design_draft.pdf.
- [53] W3C, "XML," [Online]. Available: <http://www.w3.org/XML/>.
- [54] OASIS, "eXtensible Access Control Markup Language (XACML) Version 2.0," 1 February 2005. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.
- [55] OASIS, "XACML Profile for Role Based Access Control (RBAC)," 13 February 2004. [Online]. Available: <http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf>.
- [56] OMG, "Unified Modeling Language," [Online]. Available: <http://www.uml.org/>.
- [57] V. Bahyl and N. Garfield, "DNS load balancing and failover at CERN," in *15th International*

Conference on Computing in High Energy and Nuclear Physics (CHEP 2006), Mumbai, 2006.

- [58] "Sun's XACML Implementation," [Online]. Available: <http://sunxacml.sourceforge.net/>.
- [59] "JUnit," [Online]. Available: <https://github.com/kentbeck/junit/wiki>.
- [60] "Nagios," [Online]. Available: <http://www.nagios.org/>.
- [61] "dnotify(1) - Linux man page," [Online]. Available: <http://linux.die.net/man/1/dnotify>.
- [62] A. Adeel-Ur-Rehman, F. Bujor, J. Benes, C. Caramarcu, M. Dobson, A. Dumitrescu, I. Dumitru, M. Leahu, L. Valsan, A. Oreshkin, D. Popov, G. Unel and A. Zaytsev, "System administration of ATLAS TDAQ computing environment," *Journal of Physics: Conference Series*, vol. 219, no. 2, 2010.
- [63] M. Leahu, V. Buzuloiu and D. A. Stoichescu, "A Role Based Access Control Solution for Linux Network," *The Scientific Buletin" of University Politehnica of Bucharest*, in press.

Index

A

Access Manager · iii, 6, 41, 42, 55, 59, 60, 61, 68, 70, 79, 97, 101, 103, 104, 109, 110, 122, 125, 128, 129, 140, 142
ACR · *See* ATLAS Control Room
AM · *See* Access Manager
Application Gateway · 40, 65, 66, 68, 80, 115
ATCN · 43, 45, *See* ATLAS Technical Control Network
ATLAS Control Room · 65
ATLAS Online cluster · iii, 47, 50, 58, 63, 65, 66, 68, 70, 71, 79, 80, 81, 82, 98, 102, 105, 108, 115, 117, 121, 122
ATLAS Technical Control Network · 40

C

Control Room Desktop · 39, 41, 45, 66, 115
CRD · 41, 42, 72, 73, 80, *See* Control Room Desktop

D

DAC · *See* Discretionary Access Control
Discretionary Access Control · 19

E

eXtensible Access Control Markup Language · *See* XACML

H

Hierarchical RBAC · vii, xi, xii, 24, 47

L

LDAP · 7, 45, 47, 48, 49, 50, 54, 55, 57, 58, 59, 60, 65, 68, 73, 74, 75, 76, 77, 79, 89, 95, 96, 98, 99, 100, 103, 104, 105, 108, 111, 113, 115, 116, 117, 118, 119, 121, 123, 125, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 153, 154, 156, *See* Lightweight Directory Access Protocol
Lightweight Directory Access Protocol · 7

M

MAC · *See* Mandatory Access Control
Mandatory Access Control · 21

N

NIS netgroups · 49, 50

O

Online Software · 5

P

PAP · 45, 83, 84, 86, 89, 95, 96, 97, 98, 104, 108, 154, *See* Policy Administration Point
PDP · 45, 82, 83, 84, 89, 99, 100, *See* Policy Decision Point
PEP · 45, 82, 83, 84, *See* Policy Enforcement Point
PIP · 45, 82, 83, 84, 89, 99, 100, 111, 113, *See* Policy Information Point
Policy Administration Point · 43, 45, 83, 84, 96, 97, 104, 106
Policy Decision Point · 43, 45
Policy Enforcement Point · 43
Policy Information Point · 43, 45

R

RBAC · iii, 6, 7, 9, 17, 22, 23, 24, 25, 26, 29, 30, 31, 32, 34, 35, 36, 39, 40, 41, 42, 45, 46, 47, 49, 50, 51, 53, 55, 59, 61, 72, 73, 74, 79, 80, 82, 83, 87, 88, 89, 96, 108, 111, 113, 121, 122, 127, 142, *See* Role Based Access Control
Role Based Access Control · iii, 6, 7, 9, 22, 23, 121, 125

S

Scientific Linux CERN · iii, 40, 50
SLC · 50, 55, 73, 74, 76, 82, 96, 121, *See* Scientific Linux CERN
sudo · 45, 50, 60, 73, 74, 76, 77

T

TDAQ · *See* Trigger and Data Acquisition
TDAQ Access Manager · *See* Access Manager
Trigger and Data Acquisition · 3

X

XACML · iii, 6, 42, 43, 45, 46, 61, 82, 83, 84, 86, 87, 88, 89, 93, 95, 96, 97, 98, 99, 100, 103, 105, 106, 108, 116, 123, 154