EGEE

Site Access Control Architecture

DJRA3.2

Document identifier:	**EGEE-JRA3-TEC-523948-DJRA3.2-v-1-1**
Date:	**2004-12-31**
Activity:	**JRA3**
Document status:	**FINAL**
Document link:	**https://edms.cern.ch/document/523948/**

Abstract: Virtually all actions on a distributed infrastructure ultimately result in a specific action being performed on a specific site. These actions are to be authenticated, authorized, and audited. This access control is the domain of the Site Access Control Architecture.

As such, the Site Access Control mechanisms are positioned in between global authorization (the cross domain issues and virtual organisation structures in the Grid at large) and the use made of authorization and attributes within the business logic on the services themselves. For the former, the reader is referred to the Global Service Architecture —a prerequisite for site access control— for the latter the service architecture itself is authoritative.

The security mechanisms, authentication and authorization methods, and the isolation for these tasks once inside a site, are discussed in this document.

Delivery Slip

	Name	Partner	Date	Signature
From	JRA3 (Security) project-eu-egee--jra3@cern.ch Corr. author davidg@nikhef.nl	NIKHEF, UvA, KTH/PDC, UH/HIP, UiB, and CERN (JRA1 DM cluster)	2005-01-10	
Reviewed by	Massimo Sgaravatto Christophe Blanchet Ludek Matyska		2005-01-10 2005-01-10 2005-01-10	Approved Approved
Approved by **Approved by**	PEB PMB		13/01/05 18/01/05	

Document Change Log

Issue	Date	Comment	Author
1.0	2005-01-10	Version approved by Moderator	davidg@nikhef.nl
1.1	2005-01-17	Addressed comments from Ignacio	davidg@nikhef.nl

Document Change Record

Issue	Item	Reason for Change

CONTENTS

REFERENCES

[1] Alfieri R. et al. VOMS, an Authorization System for Virtual Organizations. In *Grid Computing, First European Across Grids Conference*, 2004.

[2] Asian Pacific Grid Authentication Policy Management Authority. http://www.apgridpma.org/.

[3] The Americas Grid Authentication Policy Management Authority. http://www.tagpma.org/.

[4] Dane Skow *et al.* Site authorization service. http://www.fnal.gov/docs/products/saz/SAZ.htm.

[5] eIRG Whipe Paper Editorial Group. European Leadership in e-Science and Grids: e-Infrastructures Reflection Group Whipe Paper, The Hague edition. http://www.e-irg.org/whitepapers/.

[6] Martijn Steenbakkers et al. Guide to LCMAPS. http://www.dutchgrid.nl/DataGrid/wp4/lcmaps/edg-lcmaps_gcc3_2_2-0.0.23, and http://www.nikhef.nl/grid/lcaslcmaps/.

[7] Oscar Koeroo et al. Guide to the Job Repository. http://www.dutchgrid.nl/DataGrid/wp4/jr/, and http://www.nikhef.nl/grid/lcaslcmaps/.

[8] S. Tuecke et al. RFC3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. http://www.ietf.org/rfc/rfc3820.txt.

[9] Grid Authentication Policy Management Authority International Grid Federation. http://www.gridpma.org/.

[10] European Grid Authentication Policy Management Authority for e Science. http://www.eugridpma.org/.

[11] GEANT2 consortium. Géant2 research activities. http://www.geant2.net/ > Research > Roaming and Authorisation.

[12] Gerben Venekamp, *et al. (EGEE JRA3)* . User requirements survey. https://edms.cern.ch/document/485295.

[13] Joint Security Policy Group. Top 10 Security Requirements from the JSPG. http://agenda.cern.ch/askArchive.php?base=agenda&categ=a042157&id=a042157s4t1%2Fdocuments%2FEGEEsitesecurityrequirements-v3.1.txt.

[14] Joint Security Policy Group for EGEE and LCG. http://proj-lcg-security.web.cern.ch/.

[15] EGEE JRA3. EGEE Global Security Architecture version 1. https://edms.cern.ch/document/487004/.

[16] EGEE JRA4. DJRA4.1: Specification of interfaces for bandwidth reservation service. https://edms.cern.ch/document/501154.

[17] Kate Keahey and Tim Freeman. Workspace Management Service. http://www-unix.mcs.anl.gov/~tfreeman/workspace_service/.

[18] Linda Cornwall *et al.*, EU DataGrid Project. D7.5: Security Requirements and Test Bed 1 Security Implementation. http://edms.cern.ch/document/340234.

[19] Andrew McNab. GridSite. http://www.gridsite.org.

[20] Java.sun.com (SUN Microsystems). Package javax.security.auth. http://java.sun.com/j2se/1.4.2/docs/api/javax/security/auth/package-summary.html.

[21] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the grid: Myproxy. In *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, August 2001.

[22] IRTF AAAarch RG. RFC2903: Generic AAA Architecture. http://www.ietf.org/rfc/rfc2903.txt.

[23] IRTF AAAarch RG. RFC2904: AAA Authorization Framework. http://www.ietf.org/rfc/rfc2904.txt.

[24] Shawn Mullen, Matt Crawford, Markus Lorch, Dane Skow. Site Requirements for Grid Authentication, Authorization and Accounting (GFD.32). http://www.ggf.org/documents/GWD-I-E/GFD-I.032.txt.

[25] SURFnet B.V. The A-Select Authentication System. http://a-select.surfnet.nl.

[26] OASIS Security Services TC. Security assertion markup language (saml). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[27] EGEE Middleware Design Team. Design of the EGEE Middleware Grid Services. https://edms.cern.ch/document/487871/.

[28] EGEE NA4 Team. Application requirements database. http://egee-na4.ct.infn.it/requirements/.

[29] The OpenSSL Project. Openssl. http://www.openssl.org.

[30] University of Cambridge Computer Lab. XEN, the Virtual Machine Monitor. http://xen.sf.net/.

[31] User Mode Linux development group. User mode linux. http://usermodelinux.org/.

[32] VMWare, Inc. Vmware workstation. http://www.vmware.com/.

[33] GGF OGSA AuthZ WG. OGSA Authorization Working Group. https://forge.gridforum.org/projects/ogsa-authz.

1. INTRODUCTION

1.1. SCOPE OF SITE ACCESS CONTROL

The Site Access Control architecture describes the security mechanisms provided to assess authentication, to authorize or deny access, to provide auditing for service access, and to isolate services, service instances, and running jobs, from the system and from each other. The document's focus is towards the generic mechanisms that enable site and resource administrators to evaluate and enforce policies, and to ensure site system integrity.

The global framework for security has already been defined and described extensively[15], and this document is complementary to that global security architecture. As such, this document will not describe user- and virtual organisation related security mechanisms. Also, the document does not describe access control to objects within the services, *i.e.* the effective use of authentication and authorization data in the business logic of the service. Security functional design of the services for which access is to be controlled is left to the architecture of such components.

1.2. STRUCTURE OF THIS DOCUMENT

In this document site access control mechanisms are described in five different chapters. Issues dealing exclusively with authentication are described in Chapter 2 and subsequent local authorization decisions and the enforcement thereof in Chapter 3. Chapters 4 to 6 deal respectively with isolation, auditing & monitoring, and network access. A summary is given in chapter 7.

Each chapter provides information at three layers:

- a *Roadmap* section, putting into context the work proposed for each of the subjects, as many of the security areas see active developments outside the EGEE project. The developments inside the project have to fit in the general roadmap, even though the project itself is of limited scope and duration. In these sections, we explain how the work in EGEE relates to developments elsewhere, and how we ensure that the components developed for Site Access Control (SAC) within the EGEE project satisfy standardization efforts in GGF, and allow for convergence as specified on the *e*Infrastructure roadmap [5].
- an *Architecture for EGEE* section, describing what we envision being implemented within the planned JRA3 activity in EGEE. This focuses on the design choices made, where applicable also in perspective to the gLite architecture and design.
- an *Architecture for EGEE-1* (where relevant), describing what will be the effective architecture for the first EGEE-1 release in 2005. This architecture is of course compatible with the EGEE architecture, but implementation shortcuts have changed the detailed behaviour of the system such that not all functionality of the SAC is available.

Where relevant, additional consideration is given to components that could ease integration with security mechanisms available in many common operating systems. Although these components have no foreseen implementation within the EGEE project, we consider it important that such a backwards-compatible link is not precluded by the site access control architecture or its implementation. If and when requested by resource administrators, such components may however still be implemented.

1.3. INTEGRATION WITH OTHER AUTHORIZATION DOMAINS

Some of the policy architecture described in this document relates to the various per-user and per-VO policies that need to be enforced together with the site and domain-specific policies in a consistent way.

For this policy combination, policies and attributes will be gathered from different sources and expressed in a common language. Additionally, domain-specific policy engines can be evaluated within a common framework. The Authorization Framework, described extensively in DJRA3.1[15], allows for both use cases to be handled. These issues will be further discussed in the forthcoming update of the Global Security Architecture (DJRA3.3).

1.4. WHAT IS A SITE

The notion of *site* is usually left open for interpretation to the reader. For this document, we will use the following working definitions, taken to be compatible with its use in the GGF Site AAA Research Group requirements document[24]:

- a site is a collection of 1 or more administrative domains that belong to the same organisation. A site may (and will) have policies, and these policies prevail over any and all policies that affect the site's own resources. So, site policies restrict the freedom that the administrative domain has in determining its own policy.
- an administrative domain is a set of resources that share a common administrator, and —where applicable— a common uid/gid numbering. A set of services (1 or more) always belong to an administrative domain.

  An administrative domain has policies, and these policies take precedence over any other policies as long as these do not contradict site policies. As a trust relationship between a site and its administrative domains exists, enforcement of (global) site policies may be implemented by local resources 'calling out' to a site policy engine. Thus, the admin-domain policy enforcement acts as the source of origin of the enforcement of site policy.

The difference between a 'site' and an administrative domain only becomes relevant if the site is sufficiently large, like a large facility hosting both a supercomputer, a mass storage service, and a course-grained cluster. Each of these resources is typically managed as a separate entity, by a different set of people, and they may not share the same set of user and group numeric IDs. Still, it is a site level decision to ban a particular user from each and all of these resources (for example if that user violated local legislation). It is for these circumstances that each administrative domain policy can include a 'call-out' to a central policy decision point that inspects the user's credentials and makes a ban-or-allow decision. Such a decision is common for all administrative domains, and need not be known the domain administrators beforehand — it is enforced automatically.

1.5. REQUIREMENTS ON THE SAC ARCHITECTURE

The Site Access Control architecture described in this document is based on requirements and use cases collected from a variety of sources. These sources include the (draft) documents from the SAAA Research group in GGF[24], the EGEE Security user requirements survey[12], the EGEE NA4 application requirements[28], the prioritized list of requirements[13] from the Joint Security Policy Group[14], and prior requirements documents from the EU DataGrid project[18]. Thinking on the architectural roadmap was also influenced by the e-IRG Whitepaper The Hague version 2[5].

Note that not all the requirements in these documents relate to site access control. This document is specifically limited in scope to those issues that can and must be resolved at the site access control level by technical means. Grid-wide authorization and access control decisions, as well as issues regarding specific site and community policies are considered to be outside the scope of this document (such as

restrictions on access rights imposed by a virtual organisation upon some of its members). The architecture shown, however, provides resource owners with appropriate mechanisms to implement policies that comply with the (higher-level) requirements.

1.6. CROSS-DOMAIN POLICY CONSIDERATIONS

For effective cross-domain authorization that respects policies from the user, the virtual organisation, and the site, all these policies need to be combined to reach a common decision. Since such issues are by definition crossing site boundaries, a discussion of these issues is therefore deferred to the updated global security architecture DJRA3.3. To date, the exact semantics and behaviour of such a policy combination are still being defined.

We will need to recognise the distinction between Authentication (Subject and Identity) domains and Resource domains which may differ by policy and restriction precedence. E.g. for a subject attribute release decision (the willingness to reveal attributes related to the subject), the home (origin) policy should have precedence over proxy site/policy, but in Resource access resource policy has a precedence in access decision. For example, when a user requests a service from a site that asks the user to provide a proof of age, it is up to the user whether or not to release that information. But it is a resource policy to deny access to the user in case age information is not released. So, this is a clear example of using an ownership approach - an owner has a precedence over others. This approach is a basis for sorting out issues about policy aggregation and decision chaining. In this document we describe those policies in which the resource access policy prevails.

2. AUTHENTICATION VALIDATION

In this section we will describe the mechanisms for conveying identity assertions, the format of these assertions and the alternatives for authenticating clients to the services.

As described in the Global Security Architecture[15], we will assume the push model for sending assertions from the client to the server, since this is the best suited mechanism for grid computing as we use today. However, the assertions sent by the client will carry not only authentication information, but will also hold assertions related to authorization and policy. Such additional assertions will be addressed in the next chapter.

In the authentication roadmap for site access control we will also address some alternatives to this model, that could lead to more effective integration with alternative authentication methods, such as those used for roaming network access.

2.1. AUTHENTICATION ROADMAP

Today, grid authentication is entirely based on the push model. As part of the connection establishment (when using a transport-level security mechanism), or as part of the message (when using message-level security), all relevant user data are sent. For example, the proxy certificate[8] contains the subject name of the originator, possibly a set of attributes signed by their respective attribute authorities, and a chain of signatures relating the subject to a known root of trust at the relying parties' end. Users and relying parties both trust a common set of identity providers: the Certification Authorities (CA).

This authentication model is ideally suited for building virtual organisations where independent users join in a common venture with independent resource providers. This is also the operating model that underlies most of the grids today. All major grid infrastructures in e-Science —worldwide— can rely on a coordinated set of trust providers that comply to a set of minimum requirements (the GridPMA International Grid Federation (IGF)[9] where the EUGridPMA[10], together with the APGridPMA[2] and the TAGPMA[3] are responsible for maintaining these trust relations). The Authorities federated in the IGF issue assertions to end-entities like persons, networked entities, and attribute authorities.

The downside of this approach is that is is less suited in case larger collectives of users with common characteristics (e.g. a classroom of students) ventures on to a grid. In such cases, it is more convenient for both users and providers to be able to use the user's home organisation (UHO) ("the university") for bulk validation of the users. In that case not all students need to be issued with specific credentials or be re-registered. This of course can be done in both a push and a pull model, but in the interest of privacy preservation a pull model is often selected: the resource can query selectively for the required attributes (for which it is then up to the user to decide whether or not to give them), precluding the need for a negotiation phase[1].

In this case, the client no longer has its own set of identity assertions and only the UHO has the capability to state the user's status (in this case "student"). The client itself cannot prove this status without help. Either before authentication is attempted, or as part of the authentication validation process by the service, these claims should be attested to by the UHO, and thus the UHO must be contacted as part of the process. In case of the assertion pull model, this will require additional "callouts" by the validation process that need to contact remote sources to obtain policy information: so-called Policy Information Points. It is expected that in this scenario the authentication validation steps will be taken alongside other access control decisions by a Policy Decision Point (PDP), as part of the access control chain.

The site access control mechanisms should thus allow for both modes of authentication validation to be used. Although this requires major developments both in software and in standards definition, and is thus not realizable today, in the long term this will enable better interoperability between the existing

[1] The most known example in this area is Shibboleth: `http://shibboleth.internet2.edu/`

end-entity certificate based methods and others, e.g. those developed in the context of the Geant-2 JRA5 activity[11].

It should be carefully noted, however, that all the models described above imply that each domain of resource providers, users and identity providers (which is again a federation) will include many more trusted identity providers that what is customary today. For example in a case where large numbers of students join an infrastructure, one would have to trust each university to assert student affiliation. The level of trust placed in them, and the way they identify their subjects (students) is not uniform and difficult to audit effectively.

Thus there are important considerations regarding the risk assessment: the requirements on the trust level assured by the root authorities are intimately related to the value of the resources that they are protecting. For example, the EGEE operation today is based on the IGF trust fabric and contains many thousands of machines, several Petabytes of storage, but also expensive supercomputers, owned by organisations in many different countries world-wide. A weak authentication scheme, like one based on a valid email address only, is not adequate to protect these systems, and resource owners will not allow such weakly identified users to access the systems.

In a more dynamic federation, the quality of the (identity) assertion (information regarding its generation, the way the private data was stored, etc.) should be a part of the assertion itself, so that the authentication validation mechanism can implement constraints given the value of the resource to be protected. For example, an authority that issues certificates based on both smart-card based private keys embedded in a national password, as well as certificates that are bound to a simple key stored in a file system, should indicate the quality of the private key storage in the certificates issued. That does imply that the source of the assertions itself is trusted, but it allows that source to issue assertions with a self-chosen level of validation.

2.2. AUTHENTICATION VALIDATION ARCHITECTURE FOR EGEE

The authentication in EGEE will be entirely based on end-entity Public Key Infrastructure (PKI) using X.509 certificates and RFC3820 proxy-certificates. Also alternative systems can be deployed, provided that the result is a credential that —from the resources point of view— is syntactically indistinguishable from an X.509 (proxy) certificate.

2.2.1. STANDARD AUTHENTICATION TOOLS

The validation of authentication will be based on industry standard tools (OpenSSL[29] for C and C++ programming languages, the javax.security.auth class for Java[20]). This means that most of the authentication validation methods that are built into these can be used unaltered. Some modifications, typically those already initially provided by the Grid Security Infrastructure (GSI) from the Globus Toolkit, have been added to the default validation chain. The same software (respectively OpenSSL and javax.security.auth for C/C++ and Java) will also be used for authenticating the authorization assertions issued by an attribute authority (such as VOMS[1]).

The special extensions put onto the Transport Layer Security (TLS) protocol related to the delegation mechanism which required modifications to OpenSSL will be discontinued, given that the delegation porttype defined in DJRA1.2[27] is intended for those purposes. This means that interoperation with the standard TLS protocol is restored. This will also allow access to services via user certificates from within common web browsers. Regular browser access can then be used for authenticating to services in case delegation is not required, provided the service business logic supports web access.

From OpenSSL and Java all regular validation mechanisms are used. These include (for both providers) compulsory checking of the integrity of the certificate chain and the signatures thereon, the checking

against a locally stored copy of a certificate revocation list, and checking the certificate subject distinguished name against the namespace constraint policy.

Online checking of the certificate status via the online certificate status protocol (OCSP) for all certificates will be supported in software (for Java starting as from the adoption of Java 5). The implementation will need coordination with the identity providers to provide the status responders. This coordination is ongoing and an experimental indirect OCSP responder is operational.

Other validation methods will be implemented as a local policy decision as part of the authorization mechanism (see next section). This allows for example specifics concerning the strength of the initial authentication to be conveyed to the sites as a policy extension and a decision based thereon. This 'deferred' authentication validation can also be used to implement delayed checking of revocation information by a site-central service, or enforcement of proxy certificate lifetime constraints to a maximum of approx. 24 hours (as recommended by the GGF Site-AAA RG).

For example, it is possible to include specific X.509v3 extensions in a (proxy) certificate via a policy OID extension. That extension can be used by the credential issuer to tag that a certain certificate is linked to a secure key storage solution (smart card) from which the private key cannot be exported. Such X.509v3 extensions can be inspected and acted upon by the service as part of the policy decision chain.

2.2.2. BRIDGING SOLUTIONS

Apart from user-held credentials from which a proxy is derived by the user, MyProxy[21] and/or a service 'speaking' the MyProxy protocol can be used by the user to obtain a suitable proxy. As far as site access control is concerned, these can be treated in the same way as other credentials, but MyProxy can fulfil a role between various otherwise incompatible authentication methods as far as the user is concerned. By using a MyProxy-style repository of user credentials (access to which is in turn based on other authentication mechanisms) the different mechanisms can be bridged.

The MyProxy server can either receive existing user credentials, or request credentials on the user's behalf[2] as an Active Certificate Store (ACS). A Kerberized-CA (kCA) can generate short-term proxy credentials only (a Site Integrated Proxy Service). The latter requires that the federation be extended to include the hosting site's (Kerberos) trust fabric, but the former two solutions (traditional MyProxy and the ACS) can be used within the current grid PKI trust model. Access to the MyProxy repository can subsequently be protected by other means, such as A-Select[25] — a technology-agnostic authentication selector.

2.3. ARCHITECTURE FOR EGEE-1

2.3.1. DEVELOPMENTS FOR EGEE-1

Of the architecture described above, only part will be available for the first release of the EGEE middleware security suite. In particular, support for OCSP will be absent in this first release, as the validation hooks for OpenSSL have not been enabled, and for Java OCSP support will appear only in Java 5.

For EGEE-1 we will limit ourselves to RFC3820 proxies. To support subsequent authorization mechanisms, VOMS attributes in X.509 AC format can be embedded within them. Validation of the credentials can then take place in two ways. The basic validation of the credential is usually taken care of at the TLS handshake stage, or as part of the Java trust validation in java.security.cert.

[2] This model is used by the NERSC-CA, actually DoEGrids CA 2, where a specially modified MyProxy server acts on behalf of the user as the certificate requestor, and the user's private key never leaves the server.

More complex validation decisions, which may include the validation of CRLs, and specific authentication policies, can also be deferred to the authorization framework, provided that that framework avails over the security context data and supports appropriate validation plug-ins.

2.3.2. INTEROPERABILITY CONSIDERATIONS FOR EGEE-1

As explained in the Roadmap section, MyProxy can be used to interface with other identity and assertion providers. The interaction with the MyProxy group still needs to be established, but architecturally fitting a generic authentication interface like A-select or another Authentication and Authorization Infrastructure (AAI) onto MyProxy would enable integration of conventional grid authentication and authorization mechanisms with emerging authentication mechanisms in GN2/JRA5 and developments in the European Higher Education Area.

By using MyProxy as an intermediary, the deployment model for grid services can evolve gradually towards an integrated solution.

2.4. INTEROPERABILITY POLICY CONSIDERATIONS FOR EGEE-1

When alternative authentication mechanisms are introduced, or when key storage solutions of different strength are being deployed, machine-readable qualifiers should accompany the credential so that its relative strength can be considered for authorization purposes. For example, a specific OID embedded in the proxies issued by a particular Active Certificate Store (or other service) could be embedded, and a Authorization PDP can evaluate the OID in conjunction with the service request.

More details regarding this mechanism need to be detailed in coordination with identity providers and credential store operators.

3. LOCAL AUTHORIZATION DECISIONS AND ENFORCEMENT

The common authorization framework is described in detail in the Global Security Architecture document DJRA3.1[15]. This is a framework to be used for all grid services that is independent of the particular service syntax and the implementation language.

3.1. LOCAL AUTHORIZATION ROADMAP

At various points authorization decisions are made. They are split not only between the user, the VO and the resource, but also within the resource: some of the policy decisions could be delegated to a site-central service. Thus, also to decide on site-local (but cross-resource) policies, the authorization chains can be ordered in tree-based hierarchies, where one of the resource-local policy decision points (PDP) will make a remote invocation of another PDP to evaluate the site policy and this decision is subsequently Boolean *and*-ed with the resource-local policies. Such remote invocations can be iterative, as allowed by the Authorization Framework.

This call-out flexibility allows for incorporating other, more specialized or complex authorization frameworks, such as Role-based access control and the Generic AAA Framework[22][23]. The relationship between these and the Authorization Framework was detailed in DJRA3.1. A standard authorization callout is defined by the OGSA-AuthZ Working Group[33].

Assertions and external policies will be carried over SAML[26], either pushed into the resource as part of the message when using message-level security, as part of the security context established for a TLS connection, or retrieved within a PDP implicitly. If needed, the exact protocol to communicate with the authorization service will be defined in conjunction with other projects and in GGF. By making this service into a stand-alone entity, it can be used both from what is today the Java world (a loadable class implementing such a PDP stub) and the C/C++ world (as an LCAS module).

As indicated before, the final decision to allow or deny an access request always rests with the resource owner. The resource will have a locally determined policy, expressed in XACML, that will determine which policies to evaluate, whether and how to treat external policies (determined by user or VO and shipped to the site as part of the assertion), and which PDP to trust. Although in principle this policy could be obtained from the resource in an independent way (as an assertion signed by the resource itself), and then sent back with each service request, for all practical purposes the initial policy will be held locally by the resource in a static way (e.g. in a service-owned protected file).

The policy expression language will be XACML (or a well-defined subset thereof), with specific policy evaluations being implemented by the plug-ins. For evaluation of the policies embedded in assertions, the SAML assertions[26] will be translated into an XACML dialect for evaluation. The structure of XACML to be used for describing policies is elaborated in Appendix A.

3.2. LOCAL AUTHORIZATION ARCHITECTURE FOR EGEE

The full implementation of the vision above is subject to available project resources and to maturity of relevant standards. Thus, a migration path from the existing services and implementation towards this final vision is be defined. It does mean though that some of the functionality that is currently implemented twice (once for Java and once for C) must be refactored, but also that *ad interim* some of the proposed functionality will be implemented two times, so as to make sure that the functionality is available at all times and for all relevant languages and platforms.

The foreseen architecture and implementation is presented in the following sections.

3.2.1. AUTHORIZATION SERVICE ARCHITECTURE

For the architecture and design of the stand-alone authorization service, we still have to investigate the protocol to be used. The GGF OGSA Authorization working group[33] has written a draft document (at the time of writing in public comment period) of such an SAML-based authorization service. Another source to draw inspiration from is the FermiLab Site AuthoriZation service (SAZ) system developed at the Fermi National Accelerator Laboratory[4].

However, performance is critical: a remote authorization service puts additional demands on necessary caching and reasoning properties at the client side ('don't ask the same question again if you asked it half a second ago'). For this reason, we envision that the bulk of authorization decisions will still be evaluated locally.

The local authorization will take this into account when implementing the policy evaluation engine as a separate service. This service is a container for PDP modules that do the actual policy interpretation. Via a PDP that can recursively call other PDPs a logically single PDP can in fact be composed of a set of subordinate PDPs that together provide a common decision. The PDPs communicate using the standard authorization callout as defined by the OGSA-AuthZ Working Group[33].

3.2.2. AUTHORIZATION DECISION FUNCTIONS

To implement some of the basic site security requirements, several plug-in policy decision modules will be implemented. A large fraction of these exists already, both for C-based as well as Java-based implementations. Such modules include for example the existing ban-list and VOMS-attribute based ACLs (reworked as either XACML or GACL), but can also do some of the more complex authentication validation functions (like centralized CRL checking). New modules may become available for one flavour only, depending on site requirements prioritization.

The capability of these modules to enforce particular policies is limited by the level of detail provided in the assertions. For example, to make a policy decision of banning particular end-entities from using a service, it is required that such end-entities use their own credentials to contact a service. Whenever such details are hidden by higher-layer services (such as a job submission agent acting with an 'agent' identity on behalf of multiple users), the policy modules at the site level can no longer be used to enforce these policies.

For this architecture, we assume that the ultimate service request is always performed with the original requestors identity. This design policy decision is at the core of the site access control architecture and is supported by the general EGEE middleware architecture[3].

Every policy decision module will leave relevant data in the appropriate audit trail/log. For a policy decision point this includes the information on attributes on which the decision was based (subject name of the requestor, the wall clock time, the certificate serial number, and the like), a reference to the policy itself (when this policy is dynamically changing), on whose behalf the decision was made, and the decision reached (that is the 'allow' or 'deny' statement).

The list of policy decision points foreseen is detailed in the following subsections.

3.2.3. PDP: USER BAN/ALLOW SPECIFICATIONS

Based on a locally held list of grid user names (subject distinguished names), specific individuals can be denied access to a resource. This PDP module will be available for both C-based and Java-based services, the first being provided by the LCAS *bandb* module, and the latter in the form of the loadable *BlackListServicePDP*.

[3] c.f. DJRA1.1, section 5.1

The opposite functionality (implementing the classic 'grid-mapfile' behaviour) is also available: the Java authorization PDP being called *GridMapServicePDP*.

This PDP implements the plain-file user allow- and ban-list support known in the existing GSI implementations in the Globus Toolkit and LCG.

3.2.4. PDP: VOMS ACL-BASED ACCESS CONTROL

Based on the VOMS[1] attributes carried within the credential, access control decisions at the level of group membership, role and capability can be defined in an XACML policy. This PDP is concerned with the binary decision of allowing (or denying) access to the service as a whole — honouring fine-grained access control within the underlying service business logic (such as access to specific files or individual database objects) is within the scope of such services themselves.

The subject for these decisions is the combination of the identity (subject name), and a set of VOMS assertions. The validity of these assertions is validated by the VOMS library prior to the evaluation of the VOMS policy. In case the attributes are invalid (improperly signed or expired) they will not be considered. Depending on a locally determined policy, the entire validation may either fail, or only a subset of the attributes will be used.

The initial implementation of the VOMS ACL-based control will be based on a limited XACML subset, which will be functionally like GACL[19].

For the C/C++ based implementation, the actual policy will be expressed in GACL for the first release. LCAS, the existing VOMS-GACL system, will be used for controlling access to job submission through the Gatekeeper, next to the access control points in the Computing Element service itself.

3.2.5. PDP: PROXY LIFETIME VALIDATION

This module will test on the $(validUntil - validFrom)$ total proxy lifetime, and compare that to a site-specific maximum. Only certificates in the certificate chain that constitute proxies (either full or restricted) will be inspected, *i.e.* those certificates that are not signed by an issuer that has the BasicConstraint set to CA:TRUE.

This implements SAAARG requirement 1.4.1.1 for proxies[24]. The module is planned to appear for both the Java and the C implementations.

3.2.6. PDP: CERTIFICATE EXTENSION CHECKS

The aim of this PDP is to facilitate enforcement of SAAARG requirement 1.3.4 by allowing the policy-OID extension for each certificate in the chain to be checked against a pre-defined (site-approved) list. It requires coordination with identity certificate providers, who should embed such OIDs in the certificates they issue, so that these credentials can be differentiated by OID based on the strength of the user key storage.

At this time, identity authorities for EGEE provide a policy OID as part of the certified data in end-entity certificates (as this is highly recommended by the relevant guidelines documents). This could be broadened to other extensions as well, and in general to qualifiers for authentication (like the type of private key store associated to this certificate, as discussed in section 2).

3.2.7. PDP: PEER SYSTEM VALIDATION

With the advance of agents in grid services (that perform automatic functions not directly on behalf of a single user), there is a serious risk that unprotected authentication data will become more widespread.

Such issues have been identified already in the current infrastructure, and for some VOs jobs are typically run with the credentials of a single specific user. Although it is possible to use a deployment model in which the risk of theft of credentials is minimized, such a model is not easily enforced.

One of the afternatives would be to link a specific credential, to be used specially for these agents, to the physical machine on which the agent is supposed to run. By linking the authentication data of the agent to the domain-name of the originating system, and by implementing a PDP that checks the domain-name against the actual peer name in the security context, the abuse of such agent credentials can be limited (as per SAAARG 1.3.5.1).

Linking of the credential to the host domain name can be done either via the subject name directly, or via the SubjectAltName dNSName extension. The policy of this PDP will allow for both options in deployment.

3.2.8. PDP: CENTRAL REVOCATION CHECKING

Although in the roadmap for authentication validation, and in the Global Security Architecture, certificate status checks are foreseen to use the online certificate status protocol (OCSP), there will be a significant time span in which insufficient OCSP responders are available.

In such cases, some of the certificate status checking can be off-loaded to a central service. This relieves local systems of the requirement to periodically obtain updated CRL files, thus lightening the load on those systems as well as the load on the CRL file servers.

A PDP for certificate validation (as an implementation of authentication validation) will be implemented, that can be invoked via the authorization service (remote) interface. Thereby this PDP provides the possibility for centrally managed CRL verification, akin to what is currently available in the Site Authorization service (SAZ)[4].

3.3. ARCHITECTURE FOR EGEE-1 AND THE MIGRATION PATH

For both Java and C/C++ services, a local authorization system will be available. For Java, this will be the Authorization Framework, although only a subset of the PDPs described above will be available. For C and C++, currently the local authorization decision system is linked directly to connection acceptance, and thus source-level patching of external components is needed to incorporate the EGEE-1 authorization system. Also, in case the service is designed to use underlying operating system mechanisms for isolation of user jobs, the local credential mapping (unix UIDs and GIDs) is incorporated in the same component.

3.3.1. AUTHORIZATION FOR GRID SERVICES

For the service currently provided in a Java implementation, the Authorization Framework will support pluggable modules, that (in the future) will include the possibility to call an external authorization service. The replacement will be a gradual process, depending on progress in the definition of common semantics for the authorization interface with other efforts and in GGF, and this process is expected to converge rapidly. In any case an external authorization service will be implemented.

Such modular support is already available for C and C++ based implementations through the Local Centre Authorization Service (LCAS), although in due course this component will be obsoleted by the Java implementation in many cases.

For the job submission (and GridFTP) service, some additional re-factoring has to take place. Based on the de-facto standard Globus authorization call-out, the GT2 Gatekeeper to LCAS interface has to evolve to this new standard, and an LCAS plug-in to call the new stand-alone authorization service has to be implemented. In this way the new common framework service can be retro-fitted to the GT2 Gatekeeper.

Additional work on a new standardized credential mapping (CM) interface is needed to reach a similar level of interoperability for the user isolation mechanism. This work is currently on-going, and a first version of this credential-mapping callout specification itself could mature before the EGEE-1 release. However, by that time no implementation of this CM callout will exist, and the existing LCMAPS interface from the Gatekeeper will be used.

3.4. RETROFITTING GRID AUTHORIZATION TO EXISTING SERVICES

All grid services, both pre-existing and new, will use the local authorization framework descibed above. But many traditional services like *SSH* (Secure Shell), *CVS* (Concurrent Versioning System), and the like could also benefit from being equipped with compatible security mechanisms. Leveraging from the existing Pluggable Authentication Modules (PAM) that have become the de-facto standard on many Unices, a 'Grid-PAM' module could retrofit much of the work described above to existing systems and services. It is yet unclear if and when demand for such retrofitting emerges, but the possibility is left open in the design of the authorization services. Depending on demand and availability of resources, the actual implementation of a 'grid-PAM' module will be pursued.

3.5. DATA ACCESS AUTHORIZATION

To explain the details of the authorization for data access the steps involved to access the data are explained:

1. a File and/or a Metadata Catalog is used to resolve a *Logical File name* (LFN) to GUID
2. a Replica Catalog is used to resolve a GUID to a *Storage URL* (SURL)
3. a Storage Resource Manager (SRM) is used to resolve a SURL to a *Transfer URL* (TURL)
4. the TURL is used to access the content of the file; for example the TURL can be used in a GridFTP server.

Some of these steps could be bundled into a single step or an intermediate service could hide them from the client, but these are the most important decision points in the background.

The File, Replica and Metadata catalogs are services affiliated with a VO. The SRM and the data access (e.g. gridftp) server is affiliated with a site.

The catalog services enforce access control only over the information that they hold, but they do not (and cannot) enforce access control on the content of the files.

3.5.1. GRID ACCESS ONLY

One way of enforcing access control over data is to allow 'grid' access only. This means that users can only access their files via grid tools and services.

The easiest way of implementing this (this is also the first planned step) is to assign all the files in a storage element to a service userid, for example *gstorage* and to add a component, which runs under this identity and interacts with the user. This is the *gLite I/O* server and client.

The user uses the gLite I/O client to access files in the gLite I/O server. This server checks the user's permission in the catalogs and delivers the content from the storage element.

This solution has the advantage of avoiding proxy delegation (time consuming and poses additional risk if the delegated credential is not restricted to file operations).

The single user identity in the storage elements also significantly simplifies the transfer services, which otherwise have to implement the same proxy renewal mechanism as the computing elements.

From the site administrator's point of view this model has the disadvantage of trusting a service affiliated with a VO, the gLite I/O server. To overcome this problem this server should be deployed at each site and be run under the control of the site administrators.

3.5.2. ALLOWING LOCAL ACCESS

Allowing local access in parallel to grid access implies that the site implements mapping from grid identities to local userids. Given this, one can push the enforcement of access control restrictions to the SRM and/or[4] file servers.

In this model if a grid service has to act on the user's behalf (for example gLite I/O server or File Transfer Service), then it needs the user's credential to be delegated.

We have this model implemented in the gLite I/O components and we also have plans for the transfer services, but it adds a significant complexity in some cases (for example the transfer service has to renew the delegated credential for long transfers). The LCG Grid File Access Library (GFAL) is a client side component, so it follows this model. This model has the advantage that the file access is completely controlled by the site admins, via their mapping mechanism and permission system of their Storage Element.

This model has the disadvantage of potential inconsistencies in access control settings: according to a global catalog the user could access a file, but local settings override this, rendering a scheduled job on that site to fail. In theory one could synchronize the access control settings on replicated files in the whole distributed environment (the catalogs have all the necessary information), but we do not have plans for such a mechanism due to the differences in the access control implementations of SRMs. Currently, the SRM v1.1 specification does not include any explicit operation to manipulate permissions. Although the v2.1 interface does add some functions, only few implementations support or plan to support access control at the level of basic Unix permissions, and no implementation at this time supports POSIX ACLs[5]. Thus, synchronization of grid-level ACLs cannot be achieved in all storage locations or in a standard and consistent way.

[4] An SRM may return one-time TURLs, so no additional access control is required in the file servers.
[5] SRM v3 may allow this by adding more access control operations.

4. ISOLATION

Many of the applications using the grid today are *legacy* applications, *i.e.* applications that are designed and implemented for running by a local, trusted user on a single administrative domain. When running these essentially 'arbitrary' applications on remote resources, both users and resource owners need isolation of those applications. On the one hand, that enables resource administrators to control the behaviour of these unknown applications, and on the other hand it protects to some extent the different applications running on a shared resource from each other. The choice of isolation technology made at the resource level, should however be transparent for the higher-level authentication and authorization modules, the middleware components initiating this application, and as far as possible also for the applications themselves.

4.1. ISOLATION AND VIRTUALISATION ROADMAP

Theoretically, complete virtualisation of resources provides the best possible way of isolating applications. Such virtualisation of the resource is common in some hosting environments like Java, where the Java Virtual Machine (JVM) essentially provides a container on which limitations can be applied using Java Security policies.

Also for traditional applications, the same or a similar level of isolation can be obtained by using host virtualisation techniques like those used in Xen[30], VMWare[32], and User-Mode Linux[31]. All these provide isolation of a complete operating system environment, albeit with different capabilities and installation characteristics.

Although the JVM and Virtual Machines are a step towards a solution, it is not complete for dynamic grid applications. For example outbound network connectivity is still difficult to control, when such network connections are also legitimately used by the application in a dynamic fashion (some of these concerns are (partially) addressed in section 6.) as a generic —non-JVM specific— service. Moreover, most applications today are non-Java applications, and it is certain that Java will never be the only VM system in existence.

In practice, much of the legacy application isolation today is accomplished by exploiting traditional Unix-level security mechanisms like a separate user account per 'grid-user' or per job.

As stated earlier, which of these technologies is used to protect the host system from the guest applications, and the guest applications from each other, should be transparent.

Implementing the full virtualisation solution is beyond the scope of Site Access Control alone. Moreover, this virtualisation can be accomplished at any level: a virtual machine per application instance, a (set of) virtual machines per VO, or hosting an entire grid site on a set of only virtual machines. It is beyond the scope of this document and beyond the scope of the EGEE project to address these issues. We will assume that, to satisfy the isolation requirements of the sites in EGEE, only basic POSIX compatible mechanisms are available, running directly on top of the host systems.

4.2. ARCHITECTURE FOR EGEE

Full resource virtualisation will not be available in the EGEE architecture. Instead, isolation will be implemented using Unix accounts and the inherent separation mechanisms of the Unix operating system. Deployment of the services that require isolation will only be supported on platforms that provide a true POSIX compliant interface to userid and groupid switching.

The Unix-domain isolation will be implemented in the form of a `sudo`-style wrapper program: a lightweight code that is easily auditable at the source level, that will always execute with elevated privileges, and will subsequently change user identity. This wrapper will execute a service-determined executable

with local credentials that are derived from the user's identity and any accompanying authorization assertions (*i.e.* VOMS[1] attributes).

4.2.1. LOCAL CREDENTIAL MAPPING

Virtually all existing computer systems require that a process or action is performed using a credential. On traditional POSIX systems, this is a user ID (`uid`) and one or more group IDs (`gids`), with one specific uid and gid (the 'root' user) having elevated privileges. Other systems use authentication frameworks like AFS or Kerberos5 in lieu of, or next to the conventional POSIX authentication. Therefore, it is necessary to provide any user request that will create a process or access data directly via the file system layer with (a set of) local credentials.

Conventional GSI provides a direct one-to-one mapping between the client's distinguished name (DN) and a pre-existing local credential. Moreover, the gatekeeper service can also acquire a Kerberos ticket if so instructed by the system administrator. There is no provision either for users that are not known to the system beforehand, or to acquire privileges based on VO membership. The former point (unknown users) has been address by the 'pool accounts' extension to GSI, but this is still limited to conventional Unix credentials (uid and gid), and does not support membership of multiple VOs. But the poolaccount mechanism does allow for adding users to VO without the need for local system intervention, while maintaining tracability within the site at the per-user level.

In the EGEE architecture, the relation between the grid credentials (assertions) and the local Unix user IDs and group IDs is determined by the Local Credential Mapping Service (LCMAPS). In brief, LCMAPS is a pluggable framework like the Local Centre Authorization Service LCAS. However, in order to merge into pre-existing services that use the Grid Security Infrastructure, LCMAPS is equipped with a more advanced policy language and multiple entry interface. It can be used without recompilation or re-linking of either the Gatekeeper and GridFTP daemons. A new entry interface, that takes into account service request details, is provided for newly designed services.

More details on LCMAPS are provided in the implementation and configuration guides[6].

4.2.2. DYNAMIC ACCOUNTS AND ACCOUNTS MANAGEMENT

Additional services are needed to manage accounts obtained by using the LCMAPS or other leased account mechanisms. In particular, the obtained accounts cannot be released via grid interfaces, and their characteristics (quota, lifetime) cannot be influenced. To remedy this situation, the WorkSpace Service (WSS) is used[17]. This service is being developed in the Globus Alliance (Argonne National Lab) by Kate Keahey *et al.*

The WorkSpace Service allows a grid client to dynamically create and manage a workspace, currently implemented as a Unix account, on a remote site. The infrastructure is composed of a dynamic accounts factory (DAF) service that allows an authorized grid client to create individual accounts or groups of accounts, and a dynamic accounts service (DAS) that allows an authorized grid client to manage individual account properties, such as account access policy or time to live (TTL). These concepts are represented as WSRF services and implemented using the GT4 implementation of WSRF[17].

The code is composed of a front-end implementing protocols for the creation and management of workspaces and a back-end implementation. Workspace creation and management can be implemented in different ways according to site policies and preferences. At this point, the implementation supports two kinds of such 'back-ends': (1) true dynamic creation using the Unix 'adduser' command, and (2) account pooling implementations, one based on Andrew McNab's gridmapdir patch[19] and another based on the LCMAPS implementation.

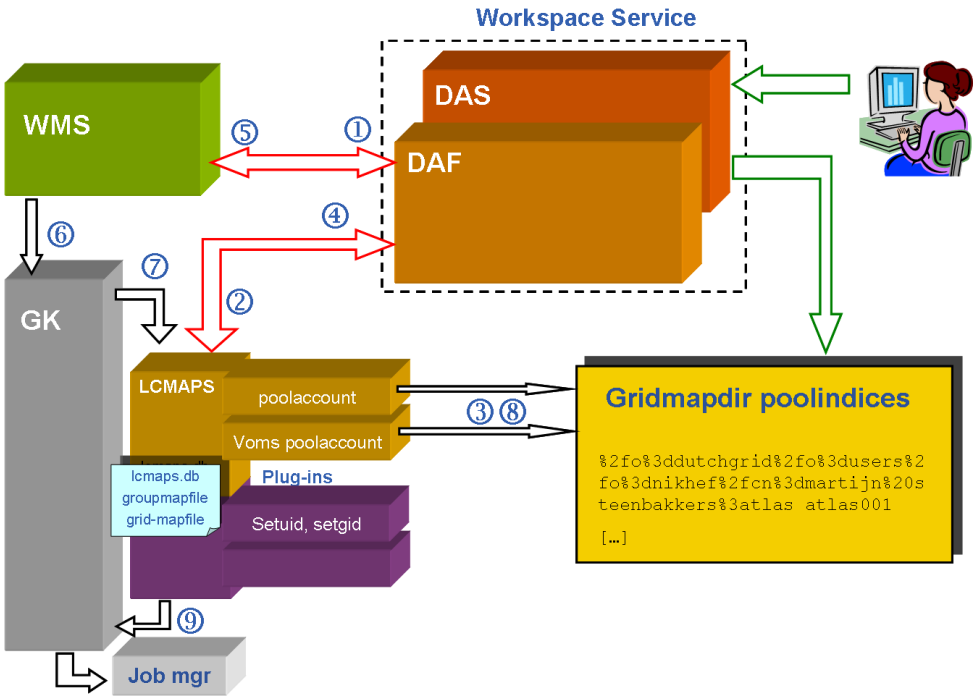

Figure 1: Relationships and invocation sequence of the Workload Management System, the WorkSpace Service, and the Local Credential Mapping Service.

The relationship between the workload management system, the DAS and the credential mapping system is shown in Figure 1.

Access to the WSS itself is again controlled by the standard mechanisms available in the Authorization Framework (this ACL mechanism will not yet be available for the EGEE-1 release). This implies that access to this service, and thus to accounts with 'special privileges', can be limited to specific groups and to users with specific roles in each Virtual Organisation. For example, only accounts associated with an administrative role (software installation of validation, for instance), can be extended beyond the site-specified default lifetime, or can avail over additional quota.

4.2.3. AGGREGATING USER CREDENTIALS

It has to be assumed that in some cases users will carry several credentials that should be equipotent when used within a service (in practical terms today: a single user holding more than one certificate with different subject names, that should all be treated as the same user). Depending on the way that the equivalence between these credentials is conveyed to the service, the isolation architecture should reflect this equipotence. In reality, though, these equivalences are most relevant to isolation of objects with a relatively long timespan, like files stored in a strategic storage element. In these cases, resolving the equipotence of the identity credentials presented can be deferred to the service implementation.

Unfortunately, this topic is still under discussion in the EGEE Middleware Security Group, pending resolution of issues regarding Data Management access control. One possible scenario implies that at the Virtual Organisation level a set of equipotent identity assertions is aggregated in a VO-level assertion (VO group membership) that can be used by the service to identify the equivalence.

4.3. ISOLATION ARCHITECTURE FOR EGEE-1

This implementation will use the modified edg-gatekeeper and the fork job-manager to implement the functionality of the light-weight `sudo` program. The WorkSpace Service will be used to manage the pool account mapping, and provide a life-time management and manipulation interface for the pool accounts, subject to appropriate authorization.

This system will thus expose two interfaces to the outside world, as shown in Fig. 1: the DAS/DAF interface (allowing for pre-selection and configuration of the account) and GRAM over HTTPS job-submission for subsequent use of this account. Only accounts that have been pre-configured explicitly via the DAF interface may be eligible for life-time extension and manipulation via the DAS. Other accounts are created without such control to preserve backward compatibility.

The invocation sequence is typically as follows: in the first stage, indicated in Fig. 1 by the red block arrows, the Workload Management System (WMS) requests a local account from the WorkSpace Service, by contacting the DAF (1) on behalf of the user. The DAF uses LCMAPS as a back-end to assign a (pool) account based on the user's DN and the VOMS attributes associated with the job (2). This assignment is registered in the so called 'gridmapdir' directory (3). LCMAPS returns the account information to the WorkSpace Service (4), which in its turn returns a handle to the WMS (5). This handle will be made available to the user and, in further releases, to specific VO members in order to manipulate the account.

In the second stage the WMS uses regular GRAM job submission (6). In this case the gatekeeper makes a call-out to LCMAPS (7), which assigns a (pool) account and switches local identity for `fork()`-style jobs (8). Control is then returned to the gatekeeper (9), that resumed the conventional flow of invoking the relevant job manager (`fork()`, `pbs`, etc.). The assigned pool account is guaranteed to be the same as the one from step (4).

The grid user that 'owns' the pool account, identified by the WSRF resource handle of step (5), can use the DAS to set the account properties, e.g the lease time, and to terminate the account. This is illustrated by the green block arrows in Fig. 1. The termination of the pool account involves a quarantine stage, in which the pool account lease is ended, but the account and its processes and files are maintained for a configurable amount of time before termination.

It is foreseen that in addition to the user 'owning' the account being able to release a mapping, also selected groups within a VO will be able to extend the account lease time of a pool account (for example, the VO software manager). Specific VO members may have the ability to remove account mappings for any VO member, but this latter functionality will not be available in the first release. The first release will also not avail over the complex ACLs to control access to the DAF: initially access will be based on subject name only, and be configured via a simple access list.

If no account pre-configuration via the DAF is done, the system will default to the traditional behaviour (in which accounts cannot be managed and mappings only forcibly expired by the site administration).

The WSS (DAF and DAS) functionality will be fully available for the EGEE-1 release, and can be called directly by qualified users (having the proper right of access to the WSS). The time of availability of management of the accounts via the Workload Management Service is still under consideration.

4.4. A SYSTEM-NATIVE INTERFACE TO MAPPED CREDENTIALS

Although auditing mechanisms as described in section 5. address the formal requirements for auditing and traceability for requests being sent to a site, these mechanisms do not always provide the most intuitive interface for resource administrators to quickly analyse a situation.

E.g., in the grid today, with the ubiquitous use of the pool account mechanism, it is hard to intuitively link a uid of a process or file to a grid entity. To ease this interpretation, the Name Service Switch (NSS) mechanism, present in most modern Unices, can be exploited. By providing a grid-mapping aware NSS

module, traditional Unix programs like `ps` and `ls` will display grid credential data when available. The type and level of detail will depend on the method of authentication, and the set of assertions passed to the credential mapping system.

Early prototyping has shown that this eases identification of grid processes on traditional Unix operating environments significantly. Depending on demand and availability of implementation resources the available prototype can be expanded and interlinked with the monitoring and (job) repository system available at the site.

5. AUDITING AND MONITORING

Any action taken on a resource via the grid interfaces must be audited and logged. This auditing information originates in various sources. Part of it is generated by the business logic of the grid services as part of their normal operations. Other elements can be logged as part of the site access control framework (the subject, target, and resource identifying information). In this section, we describe what is auditable within the site access control suite.

For EGEE-1, this auditing information will be logged via the regular service logging streams. There will be no standard format for such messages, and integrity and non-repudability of the logging stream will need to be protected by selecting an appropriate deployment model, as indicated below.

5.1. AUDITING DATA PRODUCED BY THE AUTHENTICATION PROCESS

As part of a regular authentication process (the connection acceptance and handshake), subject identity information is presented. Such information is logged by the accepting process (Gatekeeper, GridFTP service, Java TLS connection handler). The authentication information, when properly presented, is written to the service logs with the other identifying information (remote end-point and timestamp).

Actions of OCSP responders may be logged when requested, but are normally only recorded if the credential is found to be invalid. For such logging the regular mechanisms of the employed software (OpenSSL or javax.security) are used.

5.2. AUDITING DATA PRODUCED BY THE AUTHORIZATION FRAMEWORK

All policy modules (both information points and decision points) will record the information needed to reconstruct their behaviour. This includes the subject attributes, the target (resource identifier), the policy, and the decision reached. Writing such information to the relevant output streams (files, `syslog`) is left to the invoking service. Naturally, the (remote) authorization service will also log this information.

5.3. LOCAL CREDENTIAL MAPPING AND ISOLATION

All information relevant to the mapping of credentials, the credentials created or linked, and the associated attributes (VOMS groups, roles and capabilities) will be recorded by the Job Repository[7]. The Job Repository collects the user, job and user-mapping information when a job has been assigned to a fabric and handled by LCMAPS. It is a local component, and does not communicate with and is not intended to be used from outside the site.

The Job Repository consists of different parts:

- MySQL database, a database that holds information in the Job Repository database which is connectable via (My)ODBC.
- Job Repository LCMAPS module, which is loaded in the LCMAPS framework and extracts job, user and credential information storing this information in the Job Repository database.
- Job Repository API. This is a software library that acts as an interface between the LCMAPS module and the ODBC connection to the database.
- Job Repository Utilities that contain scripts to execute database creation, its recovery and dropping. It also contains a script (written in Perl with the DBI lib.) to update the Job status (to be executed by a Job Manager) and a script that can translate a FQAN in to a Unix Group ID according to its registered mapping.

By retaining a link to the local ID of the job (a batch queue system ID, or the process ID for a traditionally *fork*ed process) traceability down to the accounting system can be established.

A version of the JobRepository will be available for the EGEE-1 release.

5.4. DEPLOYMENT CONSIDERATIONS

The auditing information generated by the various components should be logged in a secure manner, on a machine different from the one on which it was generated. Traditionally, this remote logging is achieved by `syslog`, although the traditional `syslog` is highly insecure and unreliable. This could be mitigated by using secure versions of syslog, or by relying on a different remote secure logging service. In general, the security logging system should be the same as the logging system for all other services, and thus the definition and description of this service itself is outside the scope of this document. In absence of a secure logging service, it is strongly recommended that sites use a local, trustworthy solution.

The Job Repository should be deployed on or near to the systems where credentials are being mapped, as access to the backend database of the Job Repository is conducted via ODBC. The only confidential data put in the database are the authentication data, the authorization attributes, the resulting credential mapping, and the local process identifier. The Job Repository database is not used as a source of information in the authorization process.

6. SITE NETWORK CONSIDERATIONS

Both inbound and outbound network connectivity need to be controlled and authorized. A provisioning service for inbound and outbound connectivity will share the authorization framework, and will have a functional interface that is similar to the Network Element interface, as proposed by the EGEE Network Services Activity JRA4[16]. This should result in a network service that is co-allocatable with computer power and storage resources. The design of the functional part is foreseen as a co-development with JRA4. The first release of EGEE however will lack a connectivity provisioning service.

6.1. SITE NETWORK CONSIDERATIONS ROADMAP

Sites need to control their network connection from and to public areas. A site must consider the fact that their facilities can be misused to aid attacks to areas outside their domain of control. The growing user communities can introduce rogue users that abuse their privileges for launching DDoS attacks, start a *Warez share* server or make a pass-through for Worms & Viruses.

The conventional solution to this problem is to limit network access to and from the site via a firewall, and channel all permitted traffic through a set of dedicated resources (login systems, shared file servers) in a 'demilitarized zone' (DMZ), and only systems in the DMZ are connected to the global internet. Another common solution is to apply a 'default-deny' policy to both inbound and outbound traffic, except for a limited number of designated systems.

Although the potential vulnerabilities and attack patterns are the same for grid and non-grid usage, the grid has the potential to run essentially arbitrary jobs from non-local users on powerful resources inside the site. To protect the site against such arbitrary jobs, and to limit liability of the site with respect to the outside world, network isolation of the 'worker node' systems is often deemed even more important. This would translate into the requirement that a 'grid job' must be able to run without any network connectivity outside the site.

Opposite these site isolation requirements are user requirements[28][6] to have outbound connectivity to essentially any machine on the global internet, for example to access application-specific databases and metadata repositories (a biocomputing application accessing an external database accessible thourhg the web). In some cases, not having outbound connectivity will prevent the user from using any resources at a 'closed' site.

Balancing these conflicting requirements is a site issue, but it is expected that some sites can benefit from having 'managed' outbound connectivity, *i.e.* connectivity that is explicitly authorized and cleared beforehand based on local (site) policies. As such, this dynamic connectivity provisioning is similar to the provisioning of (dedicated) bandwidth to specific remote sites, as proposed for the guaranteed bandwidth service[16].

6.2. SITE NETWORK CONSIDERATIONS FOR EGEE

In EGEE we need to be able to operate in isolated non-networked environments for remotely executed jobs. In principle, only the services that are essential for the site to operate in a grid environment need inbound connectivity by default. It is also only these services that need outbound connectivity to contact other principal services at other sites. The services developed in EGEE must be able to operate fully in such an environment.

But this level of network isolation clashes with requirements from various user communities who have required for their application a certain level of outbound connectivity.

[6] Requirement R0039

The Dynamic Connectivity Service[7] (DCS) aims to provide a solution that partially satisfies both requirements by making the provisioning of connectivity explicit and policy controlled. It is then up to site policy to allow, for a particular (group of) users, and to a particular end-point, network connectivity for a specified period of time.

As most (all) sites prohibit inbound connections to a Worker Node most applications will not and should not expect this possibility from a site, although the DCS mechanism is able to provision such connections. But applications may have the possibility to connect from a Worker Node to public areas on the Internet. Today, outbound connectivity is granted by default, and sites advertise that capability in the site information system. For sites that support DCS, they should specify that capability as well, such that connectivity can be provisioned as part of a larger request to run a job.

If a site adopts total isolation as it is, some user communities will not be able to run their applications there.

6.2.1. DYNAMIC CONNECTIVITY SERVICE

The Dynamic Connectivity Service is the name for a type of service that can dynamically alter network access based on access control. The service is designed to adjust the effective firewall rules by means of a site local policy. Logically, this corresponds to altering router or firewall tables, although in any practical deployment model this role is subsumed by a dedicated system that re-routes the traffic to the internal systems affected.

We will limit ourselves to two use cases:

- Getting outbound connectivity on request by the user running on a Worker Node (Figure 2)
- Getting inbound connectivity on request by the user which is located outside a site (Figure 3). This can potentially be used to limit access even for systems offering ‘public’ services.

6.2.2. LOGICAL DESCRIPTION OF THE DYNAMIC CONNECTIVITY SERVICE

Currently there is an unlimited use of the outbound connectivity available on all deployed sites. For site security reasons it is desirable to prohibit all outbound connectivity[13], that is to systems located outside the site network perimeter. The DCS can ”punch a temporary hole” in the firewall. In Figure 2 we show the secured Web Service (WS) interface of the DCS receiving a request from the user running an application on a Worker Node inside the domain.

A request must contain the identity of the user and its available assertions to fully authenticate the identity of the requestor. The locally delegated credential (RFC 3820 proxy) of the user must be used for this authentication. The request must also contain the destination and the source of the connection to authorize an end-to-end connection, and the time period for which this connection is to be allowed. Usually that will contain two sets of IP addresses, connection protocol type (tcp/udp) and port numbers. This conforms to the set of parameters for the bandwidth agreement service[16].

The authenticated request must be must be authorized according to the site local policy on connectivity. After accepting the request it must be executed on the site local infrastructure.

A use case that has received less attention is the request for a ‘hole in the firewall’ from outside the domain. This means that the request must be able to pass the firewall through a secured socket to connect to the Web Service interface of the DCS. In figure 3 we can see that the request from outside the site will authenticate the user and trigger the DCS to authorize the user’s request for an opening in the firewalled environment that the user can use.

[7] the Dynamic Connectivity Service is also referred as Site Proxy.

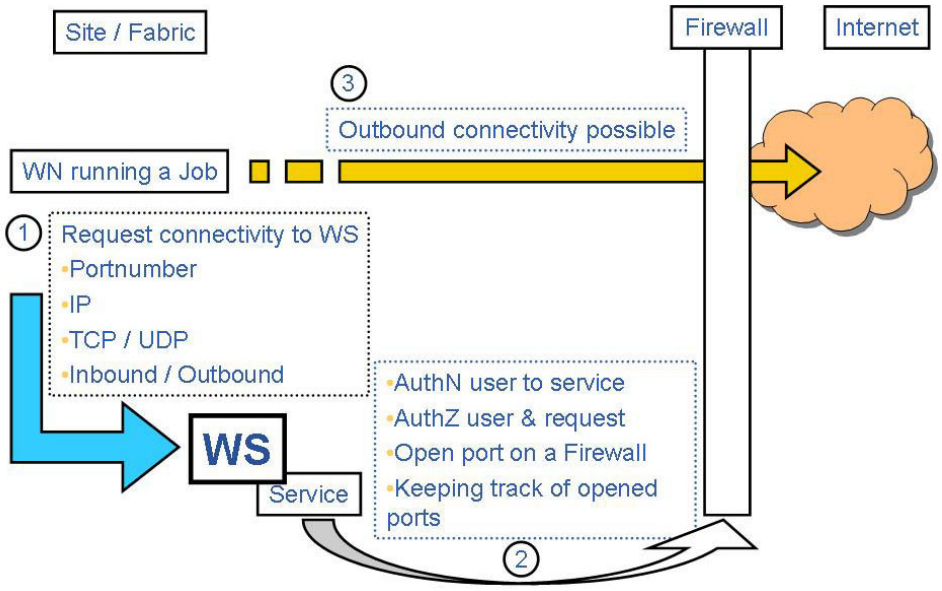

Figure 2: Dynamic Connectivity Service Use Case 1: Getting outbound connectivity from a Worker Node. The user requests, based on delegated credentials, a connection to a specified end-point outside the site (1). This request is authorized by the DCS and, if allowed, the route is enabled (2). Subsequently, the connection to the global internet is established (3), depending on the deployment model either directly or via a proxy endpoint. This graph shows the logical topology of the request. The deployment model will differ from this logical graph via the insertion of dedicated systems and indirect routing of the outbound packets via this 'proxy system'.

It is not foreseen that this will be used within the EGEE project, but the interfaces developed should be compatible with such use in order to interoperate with other *e*-infrastructures.

6.2.3. DEPLOYMENT MODELS

Every site has its own setup for network configuration and hardware. Therefore, the DCS must support as many of these configurations as possible.

There are several ways of deploying a DCS. The most straightforward deployment model is to have the DCS manipulate the site firewall or border router directly. Altering the access control lists within such a network device must be logged and controlled by a DCS service. The most likely deployment model, however, is to insert a proxy system between the source systems (worker nodes) and the firewall, and have the DCS manipulate the forwarding policies on the proxy system. That requires the proxy system itself to have unrestricted outbound connectivity, but limits exposure of the site routing and firewalling equipment.

For the latter scenario to work, the default route on the source systems (worker nodes) for accessing the global internet must be reset to the internal IP address of the proxy system. The proxy system can then do port filtering or packet inspecting depending on the site policies and DCS-configured pass-throughs, and only then forward packets to the outside. The proxy box will then have a "normal" default route. That way, the intervention of the DCS is limited to a single box, a non-critical component, and will still be backed by the regular firewall.

The last option under consideration is for the DCS interface to return a 'proxy URL' to the user, to be used indirectly to reach the specified destination on the global internet. Details are left to the networking activities SA2 and JRA4 for definition.

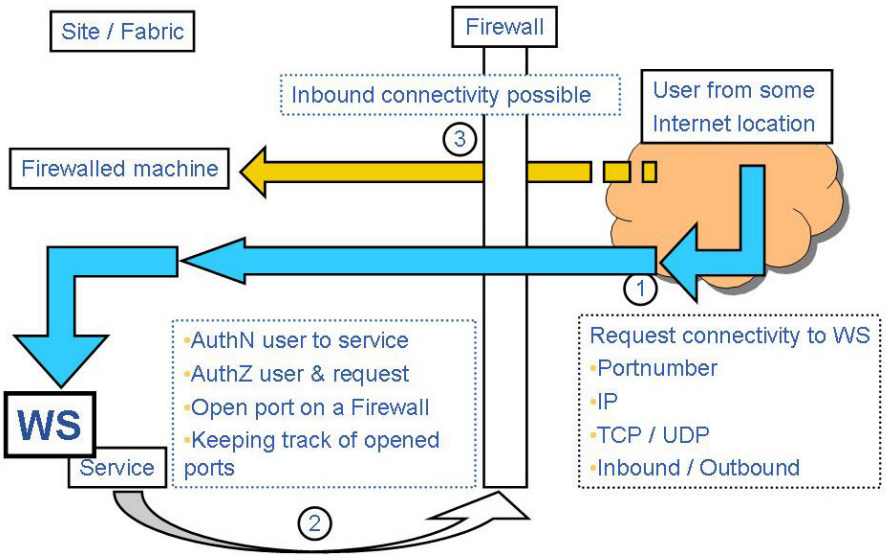

Figure 3: Dynamic Connectivity Service Use Case 2: Getting inbound connectivity from outside the site to a site-local service. In this use case a service is co-requested with network access to that service or system (1). If allowed by the site policy, network access to the machine hosting the service is granted (2), in a way that is dependent on the deployment model. Subsequently, the service can be addressed (3). This is (again) a logical description. In any real deployment model the DCS will be implemented in a dedicated system.

6.3. CONSIDERATIONS FOR EGEE-1

Currently there is a discussion group within the GGF that is investigating various solutions to network-security versus the connectivity requirements of some user communities. This discussion has not yet completed, and the DCS will be revised and extended based on the discussions with the computer-networking activities in the project.

The DCS will not be available as part of the EGEE-1 release.

7. SUMMARY AND CONCLUSIONS

This document describes several mechanisms related to site access control: authentication, policy evaluation and enforcement, isolation of tasks, the auditing of the decisions and the acquired credentials, and a mechanism to access networks in a managed way.

Many of these services can be deployed independently, are explicitly intended to be used as modular components. For example, a site that provides only a selected set of pre-installed services (like a query service for a database) can use only the authentication and authorization services, but can do without the credential mapping and dynamic connectivity services. This intentionally allows for a wide range of deployment models, and preserves site autonomy.

Also, the development area that covers security mechanisms for grid and other e-infrastructures is one of the more active in the world. Ample reference is made in this document to related developments elsewhere, and in particular in the European area. This coordination with related work will result in more flexible mechanisms and stronger security, but in due time will require changes to the site access control mechanisms.

When implementing the services described here, care will be taken so as not to weaken the functionality available at any particular point in time. Although this sometimes results in a somewhat haphazard way of installing and configuring components, we see this is as a necessary burden. Within the limited time span of the project, and with the need to support an operational infrastructure, shortcuts cannot always be avoided. In particular the isolation section (the acquisition of credentials to run a user's job on a site) suffers from this strict implementation timeline.

The focus of the mechanisms described here is also mainly oriented towards enabling site and resource administrators to implement and enforce their own policies. It is clear that in some cases, their ability to define local policies and the effective absolute power the administrators have over their resources could conflict with integrity and confidentiality requirements from users and communities. For example, the privileged systems administrator can read and possibly tamper with user data and processes. Solutions such as user-level data encryption at the source can address such issues, but their use should, by their very nature, be independent of the site mechanisms.

Any particular deployment model of these components is, by the very nature of the site security model, left up to the sites. Where relevant we have indicated likely (secure) deployment models for the specific components in the document. But this document should also serve as the basis for continuous discussion with the site responsibles to improve and extend the mechanisms to provide a secure and trustworthy grid, acceptable to resource providers and users alike.

A POLICY EXPRESSION IN XACML

A1. XACML

XACML provides a format for expressing policy for the generic Role-Based Access Control (RBAC) as used by the PDP and it defines a simple Request/Response message format. Figures 4 and 5 shows the structure of Policy element and Rule element. Policy is bound to the Target that is described by Subject, Resource and Action. Policy may contain a number of rules defined by multiple Rule elements.

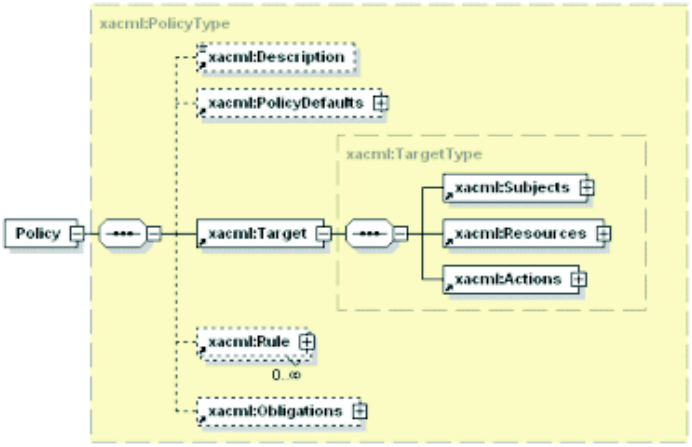

Figure 4: Definition of the Policy element in XACML binding access rules to a Target (Subject, Resource, Action).

A rule is the most elementary unit of policy. The main components of a rule are targets: conditions that are represented by subelements and effect which is included as an attribute of the Rule element.

The <Condition> element is a boolean function over subject, resource, action and environment attributes or functions of attributes. If the <Condition> element evaluates to ”True”, then the enclosing <Rule> element is assigned its Effect value. The <Condition> element is of ApplyType complex type.

The <Apply> element denotes application of a function to its arguments, thus encoding a function call. The <Apply> element can be applied to any combination of <Apply>, <AttributeValue>, <SubjectAttribute-Designator>, <ResourceAttributeDesignator>, <ActionAttributeDesignator>, <EnvironmentAttribute-Designator> and <AttributeSelector> arguments.

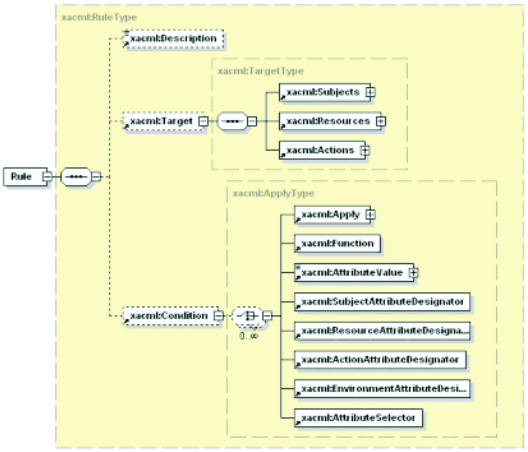

Figure 5: Definition of the Rule element in XACML defining the access Conditions to the Target (Subject, Resource, Action).

XACML re-uses enumerated list of functions and operations defined in XPath 2.0 and XQuery 1.0 used in the FunctionId attribute of the <Apply>/<Condition> element. Element Target contains matching specification for the attributes of the Subject, Resource and Action.

XAML defines format for the request message that provides a context for the policy-based decision. The request may contain multiple Subject elements and multiple attributes of the Subject, Resource and Action.

The request message consists of three mandatory elements Subject, Resource, Action (so called Target triad Subject, Resource, Action), and optionally may contain the Environment element. The Subject element normally consists of Subject attributes, Subject authentication token and may contain subject ID sub-elements. The Resource element contains ResourceID sub-element that specifies the resource or instrument, and may contain multiple ResourceAttribute sub-elements that may define resource subsystem or content related attribute. The Action element contains only one sub-element ActionID. It will be also possible to request multiple actions, however handling of such requests should be defined by the policy. The Environment element provides additional context information for the Request and can be used for Requestor's policy reference in case of mutual Authorisation.

A2. USING SAML FOR SECURITY TOKEN EXCHANGE

SAML AuthZ assertions can be used for creating AuthZ tickets, security credentials to be interpreted subject to a specific policy by a policy decision point. The use of SAML 2.0 (over 1.1) provides specific improvements that are benificial for the exchange of assertions:

1. features improving SAML security (via better integrity and secure context management):
   - Issuer element is now obligatory top level element under root element <Assertion>, it is moved from the attribute in <Assertion> element
   - <Subject> element is an (optional) top element and it is removed from the (Authn/Authz/Attribute) Statement elements as in SAML 1.1
   - main sensitive elements Subject/NameID, Advice/Assertion, AttributeStatement/Assertion now have an option of encrypted elements correspondingly EncryptedID, Encrypted Assertion, EncryptedAttribute
2. better flexibility in secure context management:
   - added new conditions OneTimeUse and ProxyRestriction instead of old DoNotCacheCondition
   - Assertions in Advice and AuthzDecisionStatement now can be referenced by also AssertionURIRef in addition to previous AssertionIDRef only
   - old element AuthorityBinding in SAML 1.1 is replaced now with new element AuthnContext that includes AuthnContextClassRef, AuthnContextDecl, AuthnContextDeclRef, or AuthenticatingAuthority
3. number of special Authentication context profiles are defined including X.509, Kerberos, PGP, XMLdsig, SSL, IP, Smartcard, mobile telephony, timesynch, etc.
4. XACML based Authorization profile is defined by introducing element XACMLAuthzDecisionStatement/Query, XACMLPolicyStatement/Query

B GLOSSARY

AA	Attribute Authority
AAI	Authentication and Authorization Infrastructure
AC	Attribute Certificate
ACL	Access Control List
APGridPMA	Asian Pacific Grid authentication PMA
AuthN	Authentication
AuthZ	Authorization
CA	Certification Authority
CAS	Community Authorization Service
CE	Computing Element
CRL	Certificate Revocation List
DAS	Dynamic Account Service
DAF	Dynamic Account Factory
DCS	Dynamic Connectivity Service
DDoS	Distributed Denial of Service
DNS	Domain Name System
DRMAA	Distributed Resource Management Architecture API
EDG	European DataGrid
e-IRG	eInfrastructures Reflection Group
EGEE	Enabling Grids for E-Science in Europe
EHEA	European Higher Education Area
EUGridPMA	European Grid authentication PMA in e-Science
FQAN	Fully Qualified Attribute Name
GACL	Grid Access Control List
GFAL	Grid File Access Library
GGF	Global Grid Forum
GRAM	Grid Resource Access Manager
GSI	Grid Security Infrastructure
GT	Globus Toolkit
GUID	Globally Unique Identifier
HTTP(S)	(Secure) Hypertext Transfer Protocol
IGF	International Grid Federation
JR	Job Repository
LCAS	Local Centre Authorization Service
LCG	LHC Computing Grid
LCMAPS	Local Credential Mapping Service
NAT	Network Address Translation
NE	Network Element
NSS	Name Service Switch
OCSP	Online Certificate Status Protocol
OGSA	Open Grid Services Architecture
OID	Object Identifier
PAM	Pluggable Authentication Modules
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PMA	Policy Management Authority (for authentication)
PKI	Public Key Infrastructure

RADIUS	Remote Authentication Dial In User Service
RBAC	Rule Based Access Control
SAAA(RG)	Site Authentication Authorization and Auditing (Research Group)
SAC	Site Access Control
SAML	Security Assertion Markup Language
SAZ	Site Authorization service (by FermiLab)
SE	Storage Element
SIPS	Site Integrated Proxy Service
SLA	Service Level Agreement
SRM	Storage Resource Manager
SURL	Storage URL
TAGPMA	The Americas Grid PMA
TURL	Transfer URL
TLS	Transport Level Security
UHO	User Home Organisation
VO	Virtual Organisation
VOMS	Virtual Organisation Membership Service
WMS	Workload Management System
WSRF	Web Services Resource Framework
WSS	WorkSpace Service
XACML	eXtensible Access Control Markup Language