GWD-I
Category Informational
GGF Working Group on
Authorization Frameworks and Mechanisms

Rich Baker, Brookhaven National Laboratory
Bob Cowles, Stanford Linear Accelerator Center
Leon Gommans, University of Amsterdam
Andrew McNab, University of Manchester
Markus Lorch, Virginia Tech
Lavanya Ramakrishnan, CNIDR/MCNC
Krishna Sankar, Cisco Systems Inc.
Dane Skow, Fermi National Accelerator Laboratory
Mary R. Thompson, Lawrence Berkeley National Laboratory
Date 2003-02-17
Revised 2003-06-06

GGF DOCUMENT SUBMISSION CHECKLIST (include as front page of submission)	
	COMPLETED (X) - Date
1. Author name(s), institution(s), and contact information	(X) – 2003-06-06
2. Date (original and, where applicable, latest revision date)	(X) – 2003-06-06
3. Title, table of contents, clearly numbered sections	(X) – 2003-06-06
4. Security Considerations section	(X) – 2003-06-06
5. GGF Copyright statement inserted (See below)	(X) – 2003-06-06
6. GGF Intellectual Property statement inserted. (See below) NOTE that authors should read the statement.	(X) – 2003-06-06
7. Document format - The GGF document format to be used for both GWD's and GFD's is available in MSWord, RTF, and PDF formats. (note that font type is not part of the requirement, however authors should avoid font sizes smaller than 10pt).	(X) – 2003-06-06

******

GWD-I
Category Informational
GGF Working Group on
Authorization Frameworks and Mechanisms

Rich Baker, Brookhaven National Laboratory
Bob Cowles, Stanford Linear Accelerator Center
Leon Gommans, University of Amsterdam
Andrew McNab, University of Manchester
Markus Lorch, Virginia Tech
Lavanya Ramakrishnan, CNIDR/MCNC
Krishna Sankar, Cisco Systems Inc.
Dane Skow, Fermi National Accelerator Laboratory
Mary R. Thompson, Lawrence Berkeley National Laboratory

Date 2003-02-17
Revised 2003-06-06

Conceptual Grid Authorization Framework and Classification

Status of This Memo

This memo provides information to the Grid community with focus on Grid security and authorization. Distribution is unlimited.

Copyright Notice

Abstract

This document specifies a conceptual grid authorization framework and classifies existing and proposed authorization mechanisms with regard to this framework. The framework is intended as the basis for future API design and standardization work.

Contents

1. Introduction.

Authorization may be perceived as a rather fuzzy term that can imply many things. This is induced by the fact that an authorization may be issued, transported, presented, verified, delegated, revoked etc. For example, an authorization may be represented by:

- a set of attributes that describes a user privilege (privilege management),
- a set of policies that defines a decision to grant access to a resource (access control)
- digitally signed material that can be used to assert access rights to a resource.

Many processes and entities may be involved in authorization. In this document we intend to consider the various authorization concepts and their relationships. We will first describe a framework that is expected to be general and abstract enough to allow a description and classification of any type of authorization system. We will start by defining some single purpose abstract entities and functions and then consider the communication sequences between these elements. Each of these elements may be mapped or combined into more specific components depending on the kind of authorization system considered.

This document will refer to other documents when concepts are considered that are not specifically concerned with authorization. For example, several security related concepts such as authenticity, integrity, confidentiality etc. will be positioned in this document but will only be briefly explained.

2. Authorization Framework concepts

This section will be concerned with explaining various basic concepts involved in authorization that will be placed in a framework. The *Authorization Framework* presented in RFC2904 and the *Generic AAA Architecture* presented in RFC2903 of the IRTF AAA Architecture Research Group and the Access Control Framework described in the ISO recommendation, ISO/IEC 10181-3:1996 I*nformation technology -- Open Systems Interconnection -- Security frameworks for open systems: Access control framework* have influenced our framework.

When reading this document one must consider the fact that the term authorization may mean one of following:

1) the process of issuing a proof of right
2) the proof of right (or reference to it) itself (an authorization token)
3) the process of checking a proof of right that is intended to grant that right (which yields an authorization decision)

To avoid this confusion one should always make a reference to the context.

2.1 The basic entities involved in an authorization.

In principle authorization decisions are made based on authorization information provided by authorities. These authorities must have a direct or a delegated relationship with either the authorization subject (e.g. user or organization member to which the authorization is issued), or with the resource that is the target of the request that prompted the authorization (e.g. owner or administrator of a resource), or with both. These relationships may be implemented using a trust mechanism based on some cryptographic method (i.e. by using some asymmetric or symmetric key mechanism) or may be implemented completely off-line (i.e. by some other trusted delivery mechanism).

This observation brings us to the definition of the three basic entities involved in authorization:

Subject: Any entity with a certain identity that can request, receive, own, transfer, present or delegate an electronic authorization as to exercise a certain right. Informally, a subject is any user of a service or resource. The subject may be identified as an individual user or as a member of a group of users. A user may also be a process that acts on behalf of a user and as such assumes some delegated form of identity. The subject may define a set of policies that determine how its authorization is used.

Resource: A component of the system that provides or hosts services and enforces access to these services based on a set of rules and policies defined by entities that are authoritative for the particular resource. Typically in Grid environments a resource is a computer providing compute cycles or data storage through a set of services it offers.

Authority: An administrative entity that is capable of and authoritative for issuing, validating and revoking an electronic means of proof such that the subject and/or owner of the issued electronic means is authorized to exercise a certain right or assert a certain attribute. Right(s) may be implicitly or explicitly present in the electronic proof. A set of policies may determine how authorizations are issued, verified, etc. based on the contractual relationships the Authority has established.

There are currently three general types of authorities in common use. **Attribute Authorities**, which issue attribute assertions that a given subject has one or more attribute/value pairs. **Policy Authorities**, which issue authorization policies with respect to resources and services offered by these resources. assertions which assert that a given subject has a certain right with respect to a given service. **Certification Authorities** (CAs) which issue certificates that assert that the entity represented by a distinguished name has the specified public key. CAs are out of the scope of this document since they enable authentication rather than authorization.

Authorization is frequently split into three distinct processes:
1) a person or organization defining an authorization policy at high-level.
2) implementing the high level policy into a certain executable form
3) the evaluation of the executable policy by a process which subsequently decides to issue a specific authorization to a subject or take a specific action.

The component performing the latter step of computing an authorization decision on behalf of the authorities is sometimes referred to as an **Authorization Server**.

Each of these three entities may implement a set of policies that determine the handling of an authorization. The policy handling function may be implemented as a hard coded piece of logic or it may be implemented by means of a flexible policy language . At this level we do not make more detailed assumptions.

2.2 Authorization sequences

Figure 1 of RFC2904 recognizes a number of basic entities that are referred to as:

1. User
2. User Home Organization.
3. Service Provider
4. AAA (Authentication, Authorization, Accounting) server

These terms can be mapped conceptually onto the entities defined above as follows:
1. User ⇒ Subject
2. The Service Provider ⇒ Resource
3. The Service Provider's AAA server ⇒ Authorization Authority.

Considering the above mapping, the authorization sequences defined in chapter 3 of RFC2904 can now be recognized as sequences between our three generic entities.

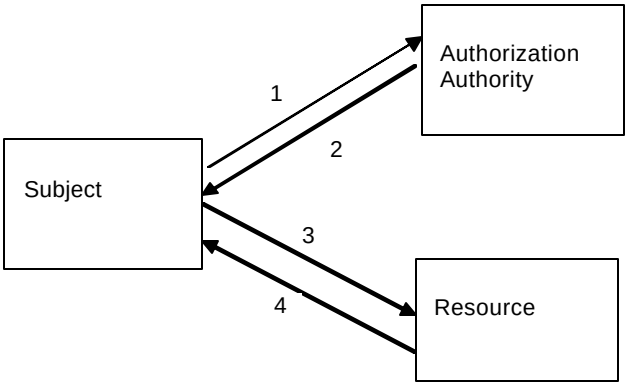

Figure 1 Authorization Push Model

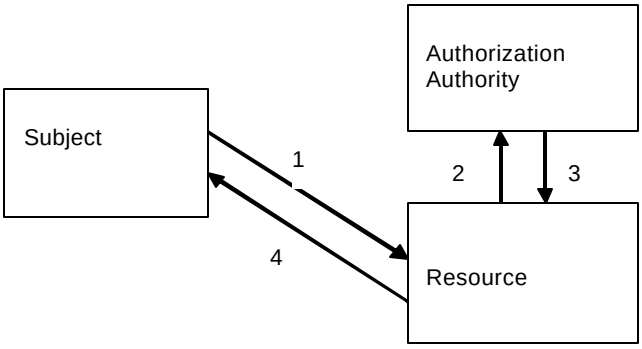

Figure 2 Authorization Pull Model

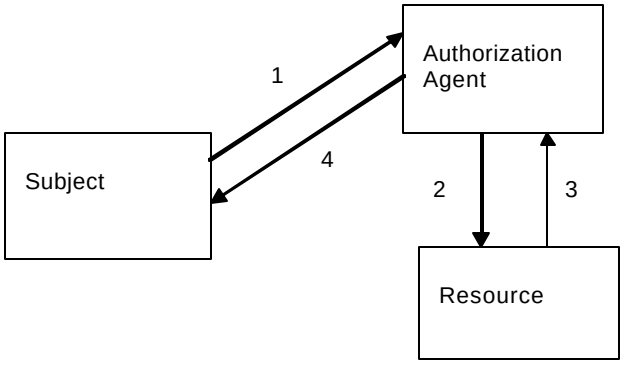

Figure 3 Authorization Agent Model

2.2.1 The authorization push sequence.

With the push sequence, the Subject first requests an authorization from an Authority (e.g. via an authorization server) The Authority may or may not honor the Subject's request. It then may issue and return some message or secured message (token or certificate) that acts as a proof of right (Authorization Assertion). This assertion should have a validity time window associated with it. The assertion may subsequently be used by the Subject to request a specific service by contacting the Service Provider. The Service will accept or reject the authorization assertion and will report this back to the requesting Subject. Examples of such sequences are found in many ticketing systems such as Kerberos or Keynote.

2.2.2 The authorization pull sequence.

With the pull sequence, the Subject will contact the Resource directly with a request. In order to admit or deny the service request, the Resource must contact its Authorization Authority. The Authorization Authority will perform an authorization decision and return a message that obtains the result of an authorization. The Resource will subsequently grant or deny the service to the Subject by returning a result message. Examples of such systems are found in the network world with systems using the RSVP or RADIUS protocol where requests typically are carried "in-band". In the Grid environment this sequence is implemented in the PERMIS and Akenti authorization systems.

2.2.3 The authorization agent sequence.

Using the agent sequence, the Subject will contact a higher level agent with a request to obtain a service authorization. This agent will make an authorization decision and if successful it will contact the Resource to provision a certain state as to enable the service. After receiving successful feedback from the service, the agent will report success to the requesting Subject. This model is relevant to Grid users when requesting a certain QoS from the Grid system (e.g. resource reservation through a scheduler). The Subject then interacts directly with the Resource to access the service.

2.3 Domain Considerations

In authorization scenarios there are at least two administrative domains: that of the Subject and that of the Service. Even in the case of access control decisions where the user is on the same machine as the resource, such as file access, there are different privilege domains. Otherwise, there would be no issue of access control. In a Grid environment the simplest case is where the Subject is in one administrative domain, its *home domain*, and the Service is in another. Another common Grid domain is a *community* or *virtual organization (VO) domain*. A VO domain can provide Authorities that allow privilege management for all the members of a VO. A common Grid scenario is one where a Subject needs to use services from several domains. Sometimes this is accomplished by a Service in one domain using a Service in another domain on behalf of the Subject. Grid Service Providers may provide services to users in multiple VO or home domains.

2.4 Hybrid models.

The three basic sequences of 2.3 are fundamental, however it does not mean that they cover all authorization situations. Sometimes, when studying a certain sequence, one may find that the framework model does not entirely match one of the above sequences. An example is the combination of the pull and push models where even though the subject has previously requested authorization from its administrative domain authority for a specific access and provides this

authorization with his access request to the resource (push) the resource may query an authority in its local administrative domain to make sure the access complies with local policies (pull).

2.5 Contractual and Trust relationships.

Contractual relationships between the domains of the different Subjects, Authorities and Service Providers are necessary to enable the acceptance and issuing of authorizations. Establishing these contractual relationships can be performed online or off-line. Once in place, they can be used as a basis for establishing trust relationships. These trust relationships may be parallel or orthogonal to the contractual relationships. E.g. a contract may provide that all involved parties should trust an independent 3rd party. Mapping of these relationships are sometimes useful to illustrate the differences. RFC2904 recognizes this in chapter 2.

2.6 Authorzation Policy and Subject Attributes.

Authorization information such as policies, attributes, identities and environmental parameters (e.g. time) are utilized and combined when making authorization decisions. Every entity may use policies to determine how a request or response should be handled. Many policies use the concepts of conditions and actions which have to be evaluated with respect to the actual request, the requesting subject's identity and the attributes this subject holds. Conditions and actions may cause message exchanges and as policies may have a certain degree of flexibility, the exchanges may not be entirely predictable. Policies may also be expressed in strings (s-expressions) that are compared and if one string (the request) is more specific than another (the policy) then the request is granted. Policies may also be hidden in the implementation of the AEF or ADF, so that certain sequences may not be possible in a given interaction.

3. Authorization Architecture

3.1 Overview

An authorization architecture consists of a set of entities and functional components that allow authorization decisions to be made and enforced based on attributes, parameters and policies that define authorization conditions. . The basic entities involved have been introduced in section 2. We will now investigate in more detail the overall architecture and focus on the information exchanged among entities and functional components. Figure 3.1 provides and overview of such an authorization system based on the pull scenario as described in section 2. It also shows various authorities that may be involved in determining authorization parameters, attributes and policies.

Similar diagrams may be drawn for the push and agent models. In the agent and push model the service request is send to the authorization server. The authorization server may have an enforcement function that drives the resource by issueing commands to the resource (agent model) or the enforcement function may issue a token to the subject (push model). The resource may have an enforcement function that will check the validity of the token (push model) or the source of the request (agent model). In the push- and agent models, subjects may also put requests to authorities that will provision the repository of the decision function. In these cases the subject is allowed to perform a certain degree of self-provisioning.

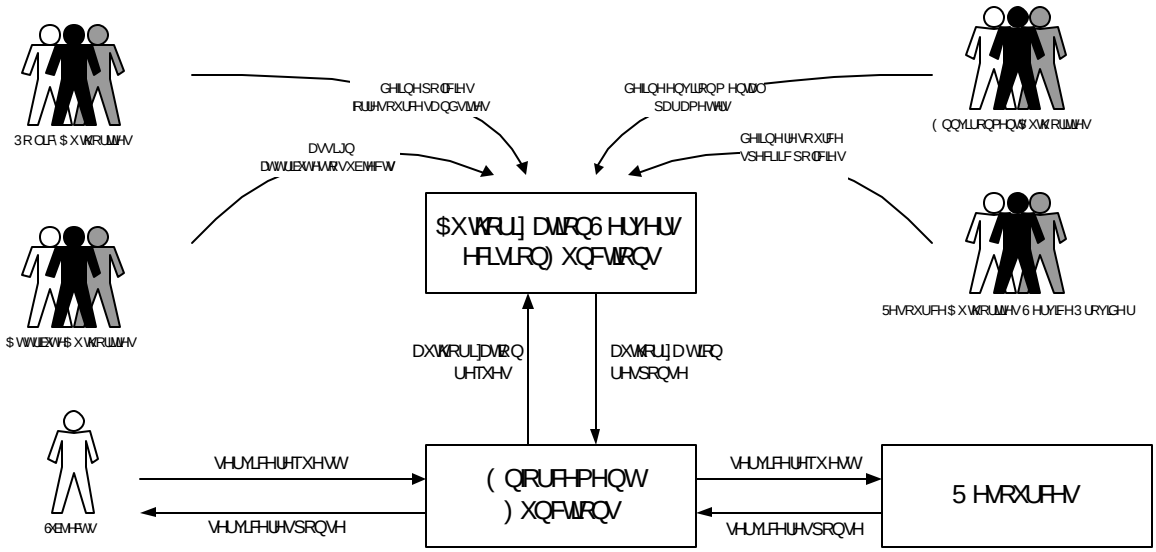

Fig. 3-1 Authorization Architecture based on the pull model.

3.2 Authorization functions

There are two access control functions defined by ISO-1011813:

Access Control Enforcement Function (AEF): Mediates access to a resource or service. It is equivalent to the Policy Enforcement Point (PEP) defined in RFC2904. It is normally contained in the Service Provider.

Access Control Decision Function (ADF): Make decisions about a subject's access to a service. It is equivalent to the Policy Decision Point (PDP) defined in RFC2904. It is normally part of an Authorization server, but can also be part of the Service Provider.

We additionally define two authorization functions to describe privilege management:

Attribute Assertion Function: issuing of an attribute assertion or attribute certificate. These assertions should have a validity period.

Policy Assertion Function: issuing of an authorization policy assertion. These assertions should have a validity period.

Other authorization functions are for example functions related to the definition of user roles and privileges. Authorization functions are embedded inside one or more administrative domains. One must be able to model how authorization functions are embedded and combined inside one or more administrative domains. In theory one can recognize the following groupings of authorization functions inside administrative domains:

1) All authorization functions are combined in a single domain.
2 a) The subject and resource functions are combined, the authority functions are independent
2 b) The subject and authority functions are combined, the service provider functions are independent
2 c) The authority and resource functions are combined, the subject functions are independent
3) all entities are independent.

In situation 2 and 3 contracts must govern the relationship and roles between the administrative domain. Specific interfaces or messages are needed to implement communication of authorization decisions between administrative domains. These interfaces must be trusted to enable reliable operation.

3.3 Flow of authorization information

Figure 3.1 simplifies the information flow in an authorization system based on the pull model. It shows attributes, parameters and policies, which are issued by the corresponding authorities, are made available to the authorization servers. The authorization servers use this information to make authorization decisions upon request by the enforcement functions.

We can define different information flow paths for authorization attributes which are also reflected in RFC3281 (Internet Attribute Certificate Profile for Authorization):

Attribute acquisition:

1. Attribute authorities provide subject attributes to the subjects (subject acquisition)
2. Attribute authorities provide subject attributes directly to the authorization servers/decision functions (server acquisition)
3. Attribute authorities provide subject attributes to repositories

Attribute application:

1. Subjects provide (a subset of) their attributes to the decision function via the enforcement function at the time of request possibly following a attribute negotiation phase (attribute push).
2. Attributes are retrieved by decision functions on demand from a repository (attribute pull)

Policies are typically stored in a repository or provisioned directly to the decision functions by the policy authorities..

3.4 Authorization information message format and exchange protocols

3.4.1 Subject Attributes

Subject Attributes need to be reliably bound to the holding subjects as well as the issuing authority. Subject Attributes must be protected to provide for integrity, issuer authoritativeness and issuer non-repudiation. This can either be accomplished by enclosing them in a digitally signed container (e.g. via an Attribute Certificate or a signed Attribute Assertion) or by issuing them over a secured channel between authenticated and trusted entities and only storing attributes in trusted repositories.

3.4.2 Authorization Requests and Responses

Authorization requests and responses are similar to attributes in that it is necessary to provide for a secure binding. An authorization request must be securely bound to a subject and the subject's service request. The authorization response must be securely bound to a request, and when required also to the response originator. In the pull and agent sequence models a request/response protocol (e.g. SAML-P, XACML) over a secured channel between the enforcement and the decision function or a digitally signed container (Attribute Certificate or Authorization Assertion) may provide this functionality. If the push model is deployed a secured container has to be used as no direct connection between enforcement and decision functions is present. If enforcement and decision functions are co-located a programming interface (AZN-API, GAA-API, PERMIS) can be used instead.

3.4.3 Policies

Policies have similar requirements to attributes as it is imperative to securely establish the authenticity of the issuer and to protect the integrity of a policy. Again a secure container or

secured connections between trusted end-points are required when policies need to be transferred or retrieved.

The policy subsystem consists of mechanisms for expressing, exchanging and processing different access control and authorization policies. Figure 3-2 shows the conceptual policy layers.

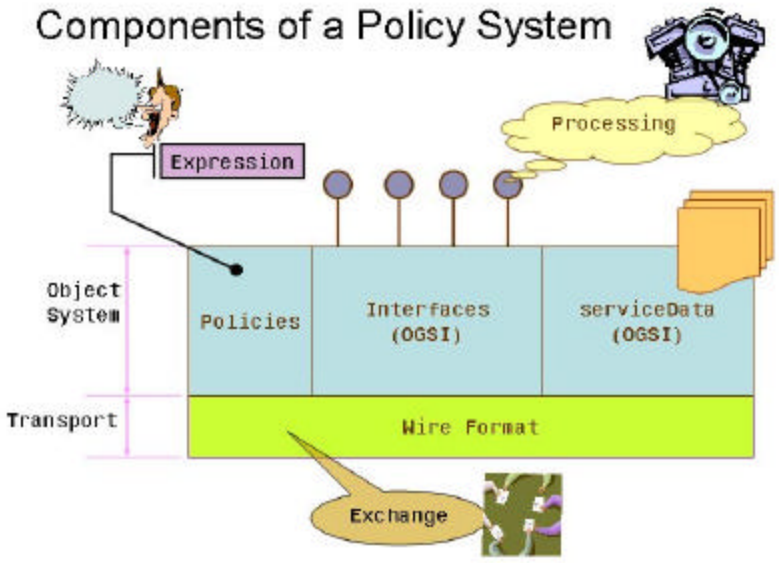

Figure 3-2 Components of a policy system – conceptual policy layers

3.4.3.1 Policy expression

The policy expression is usually done by a specific policy language which contains the vocabularies to express various policy artifacts. The language would be extensible so that domain specific vocabularies and characteristics can be incorporated in the future without fundamentally changing the language. The languages could be based on XML or s-expressions or any other language primitives.

3.4.3.2 Policy Exchange

There would be a substrate to exchange policies consisting of messages and protocols. Of course exchanging of policies would make sense only of the participating entities understand the policy expression language. Another point to note is that the exchange need not be a point to point one to one communication. There could be publish and subscribe mechanisms with associated event capturing systems.

Policy exchange also includes the exchange of metadata around policy for example the channel policies one supports including encryption, trust anchors and related information.

3.4.3.3 Policy processing

Most probably the policy processing engines would be proprietary to various systems. Existing policy systems including those based on AI and neural net paradigms can be effectively used here so long as they understand the policy expression and exchange mechanisms.

3.4.3.4 Object systems Vs wire formats

Object systems provide API compatibility between different implementations while common wire formats make it possible for heterogeneous system to talk to each other and interoperate.

For example two dissimilar policy systems can interoperate so long as they use common policy expression language and exchange the policy artifacts using common wire formats.

3.4.3.5 Policy system descriptive example - OGSA

In the OGSA world the object system would manifest itself as interfaces defined and described by WSDL. Any programming system capable of consuming the WSDL would be able to interact at the API level. The wire format in case of OGSA would be defined by the various WS-* standards. Table 3.1 shows the various policy layers in this domain. Table 3.2 shows the various proposed specifications categorized by the conceptual policy layers.

Function	Description	Standards/ Proposals
Expression	Channel / Message / System/endpoint level policies, authZ policy, Trust, QoP,… Multiple admin domains	XACML, WS-Policy, WS-XXXX, property files
Exchange	Over the wire exchange – protocols, syntax Interpretation and negotiation	SAML, WS-Security
Processing	Interfaces to understand and effect the policies	Engines

Table 3.1

Function	OGSA Security Roadmap reference
Expression	4.12.1. Grid Service Reference and Service Data Security Policy Decoration Spec GSR, serviceData
Exchange	4.7.1 OGSA trust Service Specification 4.8.1 Privacy Policy Framework Specification
Processing	4.6.1 Coarse Grained AuthZ Policy management 4.6.2 Fine Grained AuthZ Policy Mgt Spec 4.13.1. Secure Service's Policy and Processing Specification 4.13.2. Service Data Access Control Specification 4.14.1 OGSA Audit Policy Management Spec

Table 3.2

4. Framework Components

4.1 Trust Management

4.1.1 Trust Authorities

Figure 3 shows four types of authorities issuing assertions about policy and attributes in a general authorization architecture. Trust management defines these authorities and specifies what they should be trusted to do. An authority may be an individual or a group of people functioning in an administrative role, but in the architectural context, an authority is represented by some computational object, such as the owner of a file, a secure server or a public/private key pair.

Policy and resource authorities both issue policy about resources, but the policy authority operates at a higher level and may issue access control policy for a whole site or VO. It is the root of trust (sometimes call Source of Authority, SOA), and will be responsible for defining the domain's trust relationships. Attribute authorities assign attributes to subjects and may belong to the subject's domain or to a VO. Environmental authorities may define things about the resource environment, such as disk usage, or machine load, or about the distributed environment such as the security of the connection or the IP's of the client and server. Not shown in figure 3, but still present, are the authorities that establish the identities of all the entities.

In traditional systems an authority can just be a trusted file. If access control information is in the right place it is trusted by the ADF. For example, entities are defined by entries in the /etc/password file. It can also be a trusted server, such as Kerberos authenticated servers, or the NIS with which the ADF has a secured connection. In PKI based systems, the authority is likely to be represented by a public/private key pair and present its assertions in signed documents or over a SSL secured connection. At the base of such a system is the acceptance by all the participating entities of one of more CAs to verify identities

4.1.2 Defining trust relationships

Once a VO or resource domain knows how to represent various authorities, it needs to define which ones are to be trusted and for what purposes. Who defines these trust relationships is determined by the risk management strategy being used. For example, the resource provider may want the sole say on what authorities it will trust, or it may accept the decisions of a VO policy authority. In some models, the user may provide a pointer to the attribute authority that defines his attributes and the resource provider may accept it or not. The AEFs need to know which ADFs to trust for authorization decisions. In many cases these functions are collocated or have long-lived secure connections, but in some cases the AEF could trust a signed authorization assertion if it trusted the key that signed it.

Another item that comes under the category of trust management is policy about who can create proxies which have all or some the rights of the delegating entity and who can delegate rights to other entities.

4.2 Privilege Management

The term privilege in common usage refers to the list of things one is permitted to do, and implies some party willing to take action based on one's privilege. Similarly, in our discussion of privilege management, we deal with privileges (as expressed in various attributes) and assume there are policies with which those privileges are interpreted. The policies that are required to explicitly control privileges will be mentioned in this section.

Privileges can be considered a type of attribute, where an attribute is any characteristic associated with a subject that either implicitly or explicitly defines the subject’s allowed actions on

some resource. Attributes that explicitly allow some access on a grid resource are called privileges, authorization attributes or authorization assertions. More generic attributes, such as roles (for RBAC), clearance level (for mandatory access control), or group membership, may be used by an authorization server interpreting an access policy to grant the user specific actions, and thus implicitly grant actions. Typically, privilege attributes are obtained by a subject before an access to a resource and are pushed with a request associated with an authentication token (e.g. a X.509 proxy). Generic attributes may also be presented with the request as in the push model, but are often kept as part of the resource policy or in some other repository that the authorization server can query in a pull sequence.

Privilege management covers the definition, assignment, storage, presentation, delegation and revocation of both privilege and generic attributes. For management of privileges there are three distinct phases: granting the privilege, using the privilege, removing the privilege. There are two primary actors: the authority granting/removing the privilege, and the subject requesting/using the privilege.

4.2.1 Authorities

A privilege or attribute is granted to some subject by an authority for the relevant subject or attribute domain. This authority has some scope and must maintain some method of determining whether requests for a privilege come from a subject covered by this scope. Typically, an attribute or privilege authority will be in the subject's domain and will issue attributes based on policy about its known subjects. Sources of authority, their delegates and the resource domain that will accept the attribute must have a common understanding of the authority's scope. This should be expressed in a policy description, which expresses which authority may grant what attributes with what values to what subjects.

A privilege authority and an authorization server both grant actions on a resource to a subject. The distinction between the two is the domain in which they operate, A privilege authority runs on behalf of a group of users, frequently grouped into a VO. For it to be able to grant privileges, it will have had privileges delegated to it by the resource server. It will grant privileges based on policy in the subject domain. An authorization server runs in the resource domain and makes access decisions based on resource domain policy. This policy should contain information about what external privilege authorities to trust for privilege assertions.

There needs to be a method of querying the authority policy in order to determine the appropriate source of authority for a given attribute. In the push sequence, the subject will request the attribute in advance of the request, while in the pull sequence the authorization function will look for a subject's attribute in order to satisfy an access constraint.

When the subject is gathering attributes, it may know the desired end result, but not necessarily the particular attribute(s) or privilege required to achieve that result (say a file transfer from resource A to B). The discovery of the required attribute(s) or privilege is a necessary prerequisite for this step.

4.2.1.1 Managing delegation of authority

The Source of Authority may delegate portions of its authority to various agents. In general, this delegation creates a tree where the restricted authority must be expressed in some policy, the delegation must be revocable by the granting authority, and queries for relevant sources of authority should be cognizant of the delegation.

4.2.2 Privilege assignment

Privilege assignment describes the process of defining who is allowed which privileges. This includes the policy describing how a decision is made on whether or not to grant a privilege to an individual. It includes the process of assigning the privilege to an individual or group of individuals. It includes the process of suspending and/or removing a privilege from an individual or group of individuals. A privilege assertion normally contains the subject's name, the issuing authority, the name of a resource, the actions allowed on that resource and a validity period.

4.2.3 Attribute management

Attribute assertions are proofs of the right to assert a privilege. As such they have a number of common characteristics: subject, issuing authority, scope of validity, at least one attribute/value pair, and period of validity. The list of attributes one has is generated from the list of valid attribute assertions one possesses. The privilege (or actions) that the attribute allows is determined by the authorization server for the resource.

Possession of an attribute may allow the subject to act as a source of authority for that attribute within its own domain. It may delegate an attribute (limited by the policies of the source of authority for that privilege) to its own proxy or to another subject Therefore, the subject may need access to the appropriate tools to delegate and revoke its attributes The subject may define policy for the delegation, create a delegation attribute, or both. The methods of delegation must be described by the attribute authority.

4.2.3.1 Attribute Schema

Attributes are expressions of right to assert a privilege. As such there must be a common schema linking the value of an attribute to the privilege granted. This schema must be common throughout the domain of the attribute authority and authorization server.

4.2.3.2 Attribute repositories

Attribute tokens must be stored pending their use. The storage location is called an attribute repository. This repository may be under the local control of the subject or a shared facility. Subject must have access to the storage repository and must be cognizant of the policies governing access to the attributes.

4.2.3.3 Attribute assertion

The subject may choose to use an attribute in many ways. In the simplest case, attributes are attached to identity tokens in an extended x509 certificate. Since the subject may not wish to disclose an attribute when making a request which does not require it, there must be some way of knowing in advance what attributes are required to gain access to a resource. Attributes are asserted to the ADF by some protocol. This protocol may simply present the signed attribute assertions, or it may combine all of a user's attributes into a message format that it has defined. In the latter case, all the information in the assertions such as issuer and validity period must be included in or for the authorization server to trust the attributes. Tools to assert the appropriate set of attributes must be provided to the subject and all its delegates.

4.3 Policy Management

Policy is a very broad term that needs to be constrained in our context to mean access policy. Security policy may cover things outside of the authorization domain, such as standards for

message security, user identification, document encryption requirements, etc. We will limit the policy that we are interested in to information about resource access. For example, what actions are allowed by what users to what resources under what conditions? It must be possible to refer to collections of resources, actions and users in order to have a reasonably compact policy statement.

As noted under the section on Trust Management, policy is issued by policy authorities. The creation of policy frequently involves a human entity and is done in advance of the use of a resource. An ADF could query a policy authority in real time, but more typically policy will be kept in some sort of repository. This could take the form of a ACL, a data base or a collection of signed assertions. A policy repository is a natural solution for policy about static resources. Creating policy for dynamically created objects is more challenging. Sometimes it is appropriate to control access to a whole class of objects by a static policy and then just create objects within these classes. Sometimes the creator of the object may want to simultaneously create access policy for it. In this case the object creator needs write access to some trusted repository.

Some of the issues that are addressed by policy management is who can create, modify and delete policy for each resource, how quickly policy can be revoked, and where does the ADF find the policy, i.e. who/what does it trust. An additional challenge raised by distributed policy management, is how does an ADF know it has found all the relevant policy for making an access decision. One solution to this problem is to make policy only additive, so that a user starts out with no rights and accumulates them as he satisfies policy. However, this solution limits the sort of policy that can be written, especially the sort that explicitly denies access to certain individuals even though they may be a member of some group that has access.

One of the issues that needs to be addressed in policy management is how to clearly display the current policy to the resource owner or to anyone trying to add to the policy. Current Web site access rules where policy can come from several places in a configuration file and from files in a whole set of hierarchical directories, is an example of how hard it an be to figure out the access policy for a particular resource.

4.4 Authorization Context

The Authorization Context consists of those properties of the Authorization Request which are neither Authorization Privileges (such as Attributes) nor Authorization Policies (specified by or for specific resources or sites), but which are relevant to the decisions made by the Authorization server.

This includes information about the time, location, transport, and authentication of the Service Request, and may include an indication of the quality and trustworthiness of this information.

For example, a statement of the time a service request was received may also include a statement of how accurately time is established (by local hardware clock or using coordinated network time servers, for example) and also how trustworthy this statement is (for example, if the time is established using a network time server, has this been done with a secured protocol.)

Since many authorization credentials are associated with authentication credentials, the Authorization Context may also include the authenticated identity requesting the service and some statement of the strength of this authentication, in terms of numerical features (e.g. key length), protocol (e.g. PKI vs. Kerberos) and management policies (e.g. the conditions met by a Certificate Authority's practice statement.)

Similar information also applies to the transport channels used to deliver the Service Request, in terms of protocols and key lengths, for example. All of the quality and trustworthiness information in the Authorization Context may be referred to directly by requirements written into Authorization Policies, which treat aspects of the context as opaque symbols (for example, that authentication

must be by PKI certificates with a certain key length.) However, this may also benefit from a generalized way of imposing requirements on the context, including measures of quality and trustworthiness so that contexts may be evaluated numerically, rather than by simply enumerating all possible satisfactory properties.

4.5 Authorization Server

As shown in figure 3, an Authorization server is an entity that evaluates authorization requests and issues responses, taking into account relevant attributes, policies and environmental parameters. Although actual policy statements and authorization algorithms may be very application dependent, some general properties can be outlined, and the implications in terms of the policy language and algorithms can be inferred.

An authorization algorithm takes some or all of the following as inputs:

- **Nature of request** (i.e. read file, submit job, read sensor). All valid request types must be capable of being expressed in the policy language.
- **Attributes of requestor** (including delegated attributes). All attributes that will be used in an authorization decision must be expressible in the policy language.
- Attributes of resources required to fulfill request. It is not the function of the authorization server to figure out what resources are required, so if resource attributes are required by the authorization algorithm, then the required information must be supplied with the authorization request.
- **Context** (see section 4.4)
- **Environmental factors** such as system load, network load, file system available space, usage history, user quota etc. If environmental factors are to be taken into account in an authorization decision, there must exist an appropriate method to obtain the required information on a time scale that is relevant both in terms of promptness and information accuracy. The incorporation of environmental factors may be heavily application dependent, and may involve tradeoff/optimization between information accuracy and prompt decisions. The primary implications for the authorization server are that there may exist dependencies on static and dynamic environmental parameters and that the policy language must allow expression of these parameters.
- **Policy statements**. For each type of request, the policy statement must specify the criteria for issuing the appropriate authorization responses. In general, the authorization algorithm will compare the request and any attributes/environmental factors against the policy statement for that request type.

The output of the authorization algorithm is the authorization response. In many applications, a binary ALLOW/DENY response may be sufficient, but application specific languages may be developed to specify the allowed forms of response. For example, this could include a priority level or a denial that includes information about the reason.

4.6 Enforcement Mechanisms

Enforcement of fine-grained access rights is the limitation of operations performed on resources on behalf of a subject to those permitted by an authoritative entity.

In many traditional enforcement scenarios enforcement mechanisms focus on users who's authorization to access an application has to be enforced. The access of the application or service to the underlying resource (e.g. through the operating system) is of secondary concern as usually the applications are trusted software components and their system access is statically configured at the time of application deployment. In the grid context we face a different scenario. Grid applications and services can be user provided software components that are staged to the compute resources in a grid and are not necessarily trusted by the resource owners. The resources act as hosting environments for these services, which often are transient, mobile and

have to be able to migrate to a different resource if e.g. performance criteria can no longer be met. Thus, depending on the service characteristics, it may not be possible to establish static trust relationship between the service application and the underlying resource (hosting environment) up front but rather it is important to enforce not only the access of a subject to a service but also the access of a service to its current service hosting environment.

Enforcement functions can either receive the set of authorized operations up front (push scenario), or by querying an ADF (pull scenario). For this interaction a set of authorization request/response protocols (e.g. XACML, SAML-P) , which can be used to facilitate communication if ADF and AEF are remote to each other, as well as a number of programming interfaces (AZN-API, GAA-API, PERMIS), which can be used if ADF and AEF are collocated, are available.

Enforcement mechanisms can be characterized in two different groups: application dependent mechanisms and application independent mechanisms.

4.6.1 Application dependent enforcement mechanisms

Application dependent enforcement mechanisms are often directly integrated in the application or service and perform enforcement functions before the application attempts to access underlying operating system resource. This approach allows for the enforcement of very fine grained access control policies which be tailored to the specific application. Further more enforcement can be highly efficient and the service can degrade gracefully and provide helpful information to the user if a lack of authorization prevents successful completion of tasks. A drawback of integrating enforcement functionality into the application code is the need for trusted services which is an obstacle to executable staging, migration of services as commonly done in grid environments. It may also be very difficult to adapt existing legacy applications to use authorization APIs or protocols and enforce their access to resources.

For the advantages mentioned application dependent enforcement mechanisms are often favored in new service implementations and mostly applicable when services are stationary. CAS, Akenti, PERMIS leverage application dependent enforcement mechanisms.

4.6.2 Application independent enforcement mechanisms

Application independent enforcement mechanisms are separate from the service or application and take the approach of running the service in a very constrained execution environment. This permits the running of untrusted services, supports code migration and executable staging but has drawbacks with respect to the granularity of operations that can be enforced or the overhead imposed on the system, and, because of the often close ties to the operating system, the portability of systems that follow this approach. Generally two different ways to implement a constrained execution environment are prevalent:

- Operating system security functions enforced by the kernel are utilized to limit access to resources (e.g. file permissions, POSIX file system access control lists, network firewall rules, quota settings). This approach enforces access rules without performance implications and the implementation is often portable to a fair number of resource operating systems if standardized or generally adopted interfaces (POSIX) are used for the interaction. A drawback is the limitation of enforceable policies to those rules that can be translated into a security function supported by the operating system. This approach is followed e.g. in the PRIMA system.
- Resource access by the application is intercepted and evaluated before passed on to the operating system which is sometimes referred to as a "sandboxing" approach. This allows for the enforcement of arbitrarily fine grained access control rules but bears the danger of significant performance impacts due to the interception of system calls. Another drawback is the limited portability as low-level operating system interfaces are used to intercept system

calls. The Virtual Execution Environment (VXE) and Janus are typical sandboxing systems that perform system call interception. SlashGrid is also a system that layers between the application and the operating system through a virtual file system layer.

5. Classification of Existing AuthZ Mechanisms, Modules and Systems

Currently there are various existing authorization systems, mechanisms in infrastructure that are used by various grid and other applications to address the authorization concerns. We attempt here to classify these. We identified the areas of classification as

- <Infrastructure
- Authorization systems
- Components and APIs
- Policy Languages and mechanisms

We have also identified existing and ongoing work in standards bodies like OASIS, IETF and GGF to give a sense of the outreach of the effort. Detailed descriptions of these are included in the appendix.

Classification of existing systems					
	Infrastructure	Authorization systems	Components & APIs	Policy Languages& mechanisms	Standards/ Proposed standards/IETF/Oasis/ GGF
Authorization API – Generic Application Interface for Authorization Frameworks			X		
Akenti		X		X	
Cactus	X				
Community Authorization Service(CAS)		X			
Condor	X			X	
DCE		X			
EU DataGrid	X				
XACML				X	X
XrML				X	X
GAA-API			X		
ISO 10181-3 Security Frameworks for open systems: Access control framework		X			X
Legion	X				

OGSA –Security		X				X
SAML					X	X
Shibboleth			X			
PERMIS			X			
PRIMA			X			
RFC 3281: Attribute certificates and Privilege Management Infrastructure (PMI)		X				X
RFC 2704 The KeyNote trust management system			X			X
RFC2904 - AAA Authorization Framework			X			X
WS-Security	WS-Authorization					X
	WS- Policy					X

6. Security Considerations

The discussion of security concerns is intrinsic to this document.

Author Information

Rich Baker
Building 510m
Brookhaven National Laboratory
PO Box 5000
Upton, NY 11973 , USA
email: rbaker@bnl.gov

Bob Cowles (Co-Editor)
Stanford Linear Accelerator Center
2575 Sand Hill Rd., MS 97
Menlo Park, CA 94025, USA
email: bob.cowles@stanford.edu

Leon Gommans
Advanced Internet Research Group
Informatics Institute
University of Amsterdam
Kruislaan 403
1098 SJ Amsterdam
The Netherlands
email: lgommans@science.uva.nl

Markus Lorch (Editor)
Department of Computer Science

Virginia Tech (m/c 106)
Blacksburg, VA 24061, USA
email: mlorch@vt.edu

Andrew McNab
Department of Physics and Astronomy
University of Manchester
MANCHESTER
M13 9PL
England
email: mcnab@hep.man.ac.uk

Lavanya Ramakrishnan
Center for Networked Information Discovery and Retrieval / MCNC
PO Box 12889
Research Triangle Park, NC 27709, USA
email: lavanya@cnidr.org

Krishna Sankar (Co-Editor)
Cisco Systems Inc
170, W.Tasman Drive,
San Jose, CA-95134
email: ksankar@Cisco.com

Dane Skow
Fermilab
MS 369
P.O. Box 500
Batavia, IL 60510-0500
email: dane@fnal.gov

Mary R. Thompson
Lawrence Berkeley National Laboratory
MS50B-2239
1 Cyclotron Rd.
Berkeley, CA 94720, USA
email: MRThompson@lbl.gov

Glossary

Readers are pointed to the GWD-I authorization glossary document document. [glossary]

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director.

Explicit statements about IPR should not be included in the document, because including a specific claim implies that the claim is valid and that the listed claims are exhaustive. Once a document has been published as a final GFD there is no mechanism to effectively update the IPR information. Authors should instead provide the GGF secretariat with any explicit statements or potentially relevant claims.

Full Copyright Notice

References

[RFC 2704] The KeyNote Trust Management System,
http://www.ietf.org/rfc/rfc2704.txt?number=2704

[RFC2904] AAA Authorization Framework http://www.ietf.org/rfc/rfc2904.txt?number=2904

[Akenti] http://www-itg.lbl.gov/Akenti/.

[CAS] http://www.globus.org/research/papers.html#CAS-2002

[Shibboleth] http://middleware.internet2.edu/shibboleth/

[AZN-API] "Authorization API - Generic Application Interface for Authorization Frameworks",
Open Group Technical Standard C908. http://www.opengroup.org/publications/catalog/c908.htm

[GAA-API] http://www.isi.edu/gost/info/gaaapi/
[DCE]
http://www.transarc.com/Library/documentation/txseries/4.2/aix/en_US/html/atshak/atshak14.htm
#HDRHC00812

[Datagrid]
http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/Auth/VO/
http://www.gridpp.ac.uk/gridmapdir/
http://datagrid.in2p3.fr/cgi-bin/cvsweb.cgi/fabric_mgt/edg-lcfg/
http://cvs.infn.it/cgi-bin/cvsweb.cgi/Auth/voms/
http://www.gridpp.ac.uk/gacl/

[Cactus]

[SAML]
OASIS XML-Based Security Services TC (SSTC), Security Assertion Markup Language
http://www.oasis-open.org/committees/security/#documents, visited 2002-09-11

[SIG]
XML Signature Syntax and Processing, W3C Proposed Recommendation.
http://www.w3.org/TR/xmldsig-core/, visited 2002-09-11

[XACML] http://www.oasis-open.org/committees/xacml/

[WSSEC] Security in a Web Services World: A Proposed Architecture and Roadmap
http://msdn.microsoft.com/library/en-us/dnwssecur/html/securitywhitepaper.asp. Version 1.0. April
7, 2002

[RFC3281] An Internet Attribute Certificate Profile for Authorization.
http://www.ietf.org/rfc/rfc3281.txt

[Condor] http://www.cs.wisc.edu/condor/manual/v6.4/3_7Security_In.html
http://www.cs.wisc.edu/condor/presentations/PC-2002/hbwang.ppt

http://www.cs.wisc.edu/condor/classad/

Rajesh Raman, Miron Livny, and Marvin Solomon, "Matchmaking: Distributed Resource
Management for High Throughput Computing", Proceedings of the Seventh IEEE International
Symposium on High Performance Distributed Computing, July 28-31, 1998, Chicago, IL.

http://www.cs.uh.edu/~babu/poster.html
M. Lorch, D. Kafura, "Supporting Secure Ad-hoc User Collaboration in Grid Environments", accepted for publication at the 3rd Int. Workshop on Grid Computing, Baltimore, Nov. 18th, 2002
http://zuni.cs.vt.edu/publications/grid-security-V6-submission.pdf

http://zuni.cs.vt.edu/grid-security

[PERMIS] "The PERMIS X.509 Role Based Privilege Management Infrastructure" in proc. of the 7th ACM SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES (SACMAT 2002), June 2002.

[SAML] Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) Committee Specification 01, 31 May 2002, http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf

[WebSecWhiteP] Security in a Web Services World: A Proposed Architecture and Roadmap: A Joint White Paper from IBM Corporation and Microsoft Corporation , April 7, 2002, Version 1.0 availa ble from http://msdn.microsoft.com/webservices/ as the road map document

[WEBSER] http://www.w3.org/2002/ws/, http://www.webservices.org/

[XACML] OASIS eXtensible Access Control Markup Language (XACML) Committee Specification 1.0 (Revision 1), 12 December 2002
http://www.oasis-open.org/committees/xacml/docs/
s-xacml-specification-1.0-1.doc

[XRML] http://www.xrml.org/get_XrML.asp

Appendix A: Existing AuthZ Mechanisms, Modules and systems

A.1 Akenti

Akenti [Akenti] is a distributed policy-based authorization system designed for use in collaborative or Grid environments. e.g. resources, resource stakeholders, and users distributed over administrative domains. It is basically an ADF that can be run a trusted domain separate from the Service domain and exchange signed authorization assertions or can be co-located with the Service provider and called via an API. It can be called by the Subject in a push model or by the Servicer Provider in a pull model. The authorizations may be completely evaluated, or may contain some additional run-time constraints, such as machine load or disk usage that will need to be evaluated by the service provider. Akenti uses X.509 identity certificates to identify all the principals. Policy is stored as a set of XML certificates signed by the stakeholders and trusted attribute verifiers.

A.2 Attribute Certificates and Privilege Management Infrastructure (PMI)

X.509 Attribute Certificates [RFC3281] are a means to bind arbitrary attributes (e.g. role membership information, policy statements, accounting information) to identities. The binding can be accomplished by specifying either a name (e.g. X.509 DN), a X.509 public key certificate issuer and serial number, or the public key of an entity as the "holder". The issuer of an attribute certificate (AC) is often referred to as the authoritative entity. The issuer signs the AC digitally, thus ACs are self contained and do not need to be protected or stored in secured/trusted directories. ACs can be used to build a privilege management infrastructure (PMI). Issuer chains build a hierarchy similar to that in a public key infrastructure (PKI), with the root being called the source of authority (SOA). Delegation of ACs is currently not being addressed in the RFC but are possible using extensions that specify delegation rules. ACs have been deployed successfully in systems that implement discretionary access control (DAC) as well as role based access control (RBAC) mechanisms. The terms from X.509 Privilege Management Infrastructure are:

Source of Authority (SOA) = The topmost root of trust, sometimes also referred to as trust anchor
Attribute Authority (AA) (also Privilege Allocator, Authoritative Entity) = The issuer of an attribute certificate
Certificate Holder / Privilege Holder = The User or Subject of an Attribute Certificate

A.3 Authorization API - Generic Application Interface for Authorization Frameworks

The Authorization API defines a standardized API for the interface between AEF and ADF as defined in RFC2904.

A.4 Cactus

This is a problem solving enviroment in general in the high energy/ relativistic physics community. It is currently implented over Globus, so it uses GSI for security and authorization. They are currently working on Portal to launch Cactus programs that will implement a RBAC which divides users into groups with various roles that offer sets of capabilities or permissions.

A.5 Community Authorization Server (CAS)

The CAS is supporting a model of Grid usage where the users can be grouped into virtual organizations for the purpose of authorization. The assumption is that a resource provider will allow controled access to any member of a VO and then the VO can further constrain access to indvidual members of its own VO. The user wishing to use Grid resources, first goes to his VO's CAS server and gets a delegated X.509 proxy certificate that gives him the rights that the CAS

Server thinks he shold have. He then uses that credential (via GSI) to request acces tothe resources. The Resouce provider checks that it has allowed this or greater permsisions to the VO, and then grants or denies the access.

A.6 Condor

Condor implements configurable access control on all protocol operations [Condor]. Operations are categorized as read, write, administrator, config (change Condor configuration), daemon (inter-daemon protocols), owner (of the resource), and negotiator (trusted scheduler). Kerberos, GSI, NTSSPI, and Unix ident-style authentication are supported. Peers negotiate for an acceptable authentication mechanism in common. An identity map file can be used for Kerberos and GSI mechanisms. The authorization policy for each server/daemon is fully configurable (i.e., who can perform operations in each of the categories).

At a higher level, the authenticated owner of each job is included in the job's ClassAd [2] description, and resource access policies can refer to the job owner just as any other job attributes, using the ClassAd language. ClassAd Matchmaking [3] is used by the scheduler to locate compatible machines for the job and by the resource manager to ensure the job has permission to access the machine. ClassAds have also been used by others for specifying security policies (for example, in [4]).

A.7 DCE

The DCE security server is the commercial version of an authorization server derived from AFS. It uses Kerberos ids to identify users and a database of ACL's for all the resources on a system to hold authorization policy. If a user is granted access the DCE server returns a ticket that will grant access to the resource. This is a centralized scheme where one server contains all the access policy about all the resources at a site. It is implicit that all the resource gateways trust the Security server and its policy database. It is possible to set up different DCE/Kerberos realms and establish trust relations between them. In this manner a principal defined in one realm can use resources controlled by another domain. Establishing cross-realm trust turns out to be hard in practice, as is the process of administering a DCE domain in the first place. The DCE solution seems to be a bit too centralized and intrusive to be the only type of authorization acceptable by the Grid, however, it could be used by individual sites if they are willing to accept a mapping from a Grid identity to a local DCE identity.

A.8 eXtensible Access Control Markup Language (XACML)

XACML is an XML specification under development for expressing policies for information access over the internet. It "is expected to address fine grained control of authorized activities, the effect of characteristics of the access requestor, the protocol over which the request is made, authorization based on classes of activities, and content introspection... XACML is also expected to suggest a policy authorization model to guide implementers of the authorization mechanism."

While the SAML language is targeted towards expressing authentication, attribute and authorization decisions, XACML is a language designed to express the policy on which authorization decisions are based. The schema defines the elements needed to describe an authorization policy and to describe the context in which a request for authorization is made. It overlaps with SAML where it defines a protocol for
authorization requests and replies. It should be possible to define an XSLT transform to map between SAML and XACML request and reply schema. The target to which a policy applies is a subject requesting a set of actions on a resource. All three elements can be specified by attribute designators thus allowing a policy to apply at a variety of scales. For example the subject can be users who have the attribute role = adminstrator; the resources can be the top of a hierarchy and

there can be a set of actions. Usage policy is written in terms of rules which specify conditions under which a target is either allowed or denied. The rules are combined using a rule-combining algorithm into policies which include obligations that must be met when the actions are performed. XACML obligations are comparable to the conditions on SAML assertions. The polices can then be combined into policy sets to allow different types of policies, or policies written by different stakeholders to be combined.

XACML also defines elements to express the context of the user who is requesting an action. An XACML request element contains the same subject, resource and actions elements as a SAML authorization request, but adds an optional environment element. The environment can contain SAML assertions about how this user was authenticated and what attributes he might have. The XCML response contains a lists of results which are decisions relating to one resource, each of which may have obligations attached. So far the schema does not explicitly add digital signatures but recommends use of the XML Digital Signature Syntax standard from W3C 3240 if any of the information is passed over non-secured communication channels.

A.9 eXtensible rights Markup Language (XrML)

eXtensible rights Markup Language is a language to describe the rights and conditions for using digital resources.[XRML] It has it roots in the Digital Property Rights Language introduced by Xerox PARC in 1996. It has been developed commercially and has selected as the basis for MPEG-21 REL (Rights Expression Language) proposed standard. It has also been submitted to OASIS Rights Language Technical Committee as a foundation for the development of a rights language. The key top-level element defined by XrML is a licence. It contains one or more grants which define the rights of a user to a digital resource with some optional conditions applied. There are documents defining core types of resources, rights and conditions and standard extension types. The language is also designed to accepted custom extensions to all three elements.

An XrML license is comparable to a SAML assertion and there is work going on in the Web Services community to describe a general framework to enable XML-based security tokens to be used with WS-Security. Two profiles that use this general framework are provided: one for SAML and XrML.

A.10 EU DataGrid Virtual Organisation and local Access Control

This consists of five components, three in production and two as advanced prototypes available from the EDG software repository:

An LDAP-based VO service, listing Distinguished Names of members of VO's and subgroups within VO's. These are filtered using standard LDAP client tools to produce local grid-mapfiles or other local sources of authorisation information. About a dozen VO's participating in the EU DataGrid Testbed are currently in production, with several hundred users in total.

Pool account extensions to Globus gatekeeper and GridFTP server (and other service that use the Globus gss_assist library) This allocates an existing pool account to each new user of the site that connects via a Globus server. Local access control is then done in terms of standard Unix user and group permissions associated with the account when it was created. This complements the LDAP-based VO service which cannot easily know local Unix account names at each testbed site.

Local Centre Authorisation Service extension to Globus gatekeeper etc - a modular framework allowing authorisation decisions to be made on the basis of GSI proxy names as well as job description and available timeslots.

A prototype is available for a Virtual Organisaton Membership Service, which can be contacted

with a GSI proxy to request additional signed credentials giving VO and group membership, and roles within them. A client tool allows these credentials to be included as extensions to GSI proxies, in such as way that VOMS-aware services can recognise them, but unmodified Globus services continue to work.

Prototypes are also available of GACL, a library for parsing local access control lists in XML in terms of GSI DN, VOMS or CAS groups/roles or LDAP VO memberships. This is used by the EDG Storage Element (mass storage server), SlashGrid filesystem framework (to provide local disk storage controlled by grid credentials) and GridSite (which provides HTTPS file serving controlled by grid credentials.)

A.11 GAA API - Generic Authorization and Access Control API

The IETF proposed standard Generic Authorization and Access control API (GAA API) (http://www.isi.edu/gost/info/gaa_api.html) is a definition of a simple interface between a resource gateway and an authorization module or server and a policy language in which stakeholders express their access requirements to the authorization server. The API consists basically of three calls: one to return the policy associated with a resource, one to take that policy as input, along with the principal's identity and the access required and return a yes, no or maybe answer, and one that takes the principal and the resource name and returns all the rights of that principal. The get-policy call lets the caller specify a pointer to the policy data base as well as to provide an upcall to return any dynamic policy. The maybe return of the check-authorization call, allows the gateway to do further checking based on data that only it knows. The policy language consists of an ordered list of three value tokens: type, defining authority and value. These tokens can express permissions, principals defined by various identifiers, e.g., X.509 or Kerberos, trust relationships, e.g., who can authorize a token, and conditions on permission such as time, location, message protection, quotas and subject attributes. The language can also express capabilities and can handle credential delegation. So far there has not been a lot of implementations of this interface, but Globus GIS has released something recently.
GAA provides an uniform interface that various gateway servers can call. It implies that there can be distributed and local policy information but managing this information is up to the implementation. The type and format of the policy information is left up to the implementation as well, as long as it can be translated into the GAA policy tokens. GAA allows for considerable interaction between the resource gateway and authorization server at time of attempted resource access.

A.12 ISO 10181-3 Security Frameworks for open systems: Access control framework

In ISO 10181-3 [ISO 10181-3] similar components of an authorization framework are defined as follows:

PEP = Access Enforcement Function (AEF)
PDP = Access Decision Function (ADF)
Policy = Access Control Information (ACI)

[ISO 10181-3] ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996 "Security Frameworks for open systems: Access control framework"

A.13 Legion

In Legion, everything is represented as an object and objects communicate via method calls. Therefore, access is defined as the ability to call a method on an object. Access control is not centralized in any one part of the Legion system. Each object is responsible for enforcing its own access control policy (perhaps by interacting with other objects). The general model for access

control is that each method call received at an object passes through a "MayI" layer before being serviced. MayI decides whether to grant access according to whatever policy it implements. If access is denied, the object will respond with an appropriate security exception, which the caller can handle any way it sees fit. The principals in the allow and deny lists may specify particular users, the calling object's class, or the calling object itself. Users can also easily construct groups (as just another object), and define authorization policies based on these groups. This is a bit similiar to the CORBA and Java security mechanisms where each method can make a callout for authorization before it starts to execute.

A.14 OGSA Security Architecture

It is the intent of the developers of the Open Grid Services infrastructure to incorporate the Web Services Security schema elements into the service interfaces, thus providing a common way for clients and services to communicate security requirements and the tokens necessary to enforce these requirements. At that point, it may be possible to write an authorization server that can be used by multiple PEPs. If the policy that such a server uses is formatted in a standard way such as XACML policy statements, it may be possible for a resource site to use a standard authorization server and just customize the policy to meet its requirements. At the point when robust, open source Grid middle-ware is using a standard authorization and trust vocabulary and authorization servers to interpret this vocabulary exist, authorization will be at the point that authentication was when the first open source implementation of SSL and then GSI was released. This is the point when we are likely to see a standard Grid authorization mechanism.

A.15 PRIMA - Privilege Management and Authorization Services

PRIMA [1] is a privilege management system that employs X.509 attribute certificates to bind rights to users (or their surrogates). Users and administrators alike can manage and delegate privileges directly through the creation and exchange of attribute certificates. Delegated privileges can be combined and applied selectively to a specific request providing for least privilege access. This enables ad-hoc collaborative groups to exist without the need for a centralized management. Fine grain privileges are enforced on the resources through commonly available POSIX 1E file system access control lists. This provides for fast and reliable access control that fully supports legacy applications and services.
PRIMA user tools for privilege management and a proprietary resource gatekeeper are currently implemented in Java. A C implementation that can interoperate with the Grid Security Infrastructure and the Globus gatekeeper as well as a dynamic user account management system that mitigates the requirement for user accounts on every resource are under development and will be released shortly [2].

A.16 PERMIS X.509 Based Privilege Management Infrastructure

PERMIS [PERMIS] is a privilege management infrastructure (PMI) that complies with the authorization framework defined in [RFC2904]. PERMIS uses a role based access control (RBAC) scheme that allows privileges to be associated with roles rather than with specific entities. Entities can then hold several roles. X.509 attribute certificates to securely bind and communicate role membership among the components of the infrastructure. Access requests are evaluated via an authentication, authorization and accounting (AAA) server which provides two main functionalities: an access control decision function (ADF) and an access control enforcement function (AEF) as defined in ISO/IEC 10181-3. The functions interact through a PERMIS Java API. LDAP repositories are employed for credential storage. A RBAC policy language based on XML has been developed for the PERMIS PMI.

A.17 RFC 2704 The KeyNote trust management system

RFC 2704 defines trust management as a unified approach to specifying and interpreting security policies, credentials, and relationships. It defines a simple language consisting of "Fields" and values to express both authorization policy and credentials. The credentials and access rights are local to an application scope. The application provides a compliance checker that understands the rights, resource and credentials.

A.18 RFC2904 - AAA Authorization Framework

In RFC2903-2906 an “administrative domain" defines a set of users and/or resources that have a common administrative entity that defines access permissions. RFC2904 focuses on four elements of an authorization framework: The user; the User Home Organization (UHO), which, based on a user agreement, can check if a user’s request for a service should be permitted; the service provider containing an AAA server, which (coarsely) authorizes access based on an agreement with the UHO, without knowing about the individual users; and the service equipment that provides services

For evaluation and enforcement of access policies RFC2904 introduces a Policy Decision Point (PDP) and a Policy Enforcement Point (PEP). A PDP makes access control decisions based on policies defined by parties authoritative for the PDP's administrative domain. The policies can either be stored in repositories or supplied through secure messages (such as attribute certificates). PDPs can be located at the UHO and the service provider's AAA server. A PEP is the corresponding entity, typically located at the service equipment, that can enforce an access decision made by the PDP.

Two operational models for the PEP are:
A PEP queries the corresponding PDP to make an access decision when it receives a request.
A PDP provisions a set of enforcement rules to the PEP before requests arrive.

Between authorization entities (UHO, service provider AAA server, service equipment, user) authorization tickets are exchanged. Such tickets can be implemented using attribute certificates (proposed in RFC2904).

A.19 Security Assertion Markup Language (SAML)

SAML [SAML] defines a language and protocol to exchange authentication and authorization information. Its primary goal is to provide a mechanism by which permissions management data can be shared in a standardized fashion across domains and across a variety of systems and thus provide for interoperability. Among other scenarios, this enables single sign-on functionality for distributed systems. Its prime focus are web services. A standard XML message format for SAML requests, assertions and responses is provided

SAML assertions may be used within a proprietary request/response protocol. The XML Signature Specification [SIG] is suggested as an XML compliant syntax for signing SAML messages and assertions. Other mechanisms, such as secured channels (e.g. TLS) may be used instead of XML Signature to provide for message integrity, authentication of message issuer and message origin and for non-repudiation.

SAML provides schemas for queries and replies and for security related assertions. Queries can ask about a subject's authentication, a subject's attributes, or a subject's authorization to access a resource. Responses contain a status message and an assertion. Assertions may contain conditions and advice, will contain an assertion statement and may be digitally signed. Each type of statement contains a subject element which supports a variety of name types including a name identifier, security domain, confirmation method and optionally a public key of the subject. The conditions contain run-time constraints that must be evaluated by the PEP. Some examples of conditions are time periods, audience or target restrictions.

The assertion statement is one of: authentication statement, attribute statement or authorization statement. An authentication statement contains the method that was used to authenticate the user, the time at which it was done, and optionally the locality for the entity that was authenticated. An authentication statement can be used between an authenticating agent and a PDP or PEP.

An attribute assertion states the values that a subject has for a set of attributes, which are defined by attribute name and namespace. This assertion serves the same function as the X.509 attribute certificate. A SAML attribute assertion is written in XML, is optionally signed and the issuer can be identified by a variety of types of user names. The X.509 attribute certificate is written in ASN.1 notation and is always signed by an issuer defined by an X.509 distinguished name. It can contain optional extensions to help verify the holders identity, to target these attributes to a limited audience and to carry community defined information. Either certificate can contain multiple attributes each with multiple values for a single subject.

An authorization assertion supplies a statement that the request for access by the subject to a resource has resulted in the specified decision based on some optional evidence. The decision values are permit, deny or indeterminate. In the indeterminate case, the assertion may request additional information about the subject's attributes. The optional evidence consists of a list of the assertions that the issuer relied on when making the authorization decision. Since SAML authorization assertions always have an issuer and issue time, and may be digitally signed and have a validity period, they can be used as digitally signed capability certificates. In some Grid applications it is useful for a requestor to go to an authorization server and get an authorization ticket in advance of his use of a resource. When this ticket is presented to the PEP, the PEP needs only to confirm that the presenter of the capability certificate is the holder and verify that the certificate is signed by a trusted third party (the PDP). The advanced reservation scenario is one example where capability certificate (in the requestor's name) could be obtained by the scheduler and then presented by the requestor when he claims the reservation.

A.20 Shibboleth

Shibboleth, a project of Internet2/MACE, is developing architectures, policy structures, practical technologies, and an open source implementation to support inter-institutional sharing of web resources subject to access controls. In addition, Shibboleth will develop a policy framework that will allow inter-operation within the higher education community. Access control in Shibboleth is based on attribute assertions made by the user home organization, encoded in SAML.

A.21 Web Services Security

Another recent attempt to provide a standard XML vocabulary for defining the entire range of security issues in distributed systems is taking place within the Web Services community. Web services are an approach to building distributed applications is a loosely coupled fashion inspired by the model of Web clients (browsers) and Web servers. The clients and web services communicate via SOAP messages which carry XML formatted information. Since Web services communicate via SOAP messages and are independent of transport protocols, higher-level services such as security need to be defined in a message-centric, transport-neutral way. Thus each of the security specifications is expressed in an XML schema which defines meta-data that can be carried in a SOAP message header.

The WS-security framework defines seven specifications of security functionality. They are WS-security for single message security, WS-policy for security policy, WS-trust for trust relations, WS-SecureConversation for secure session connections, WS-Privacy for expressing privacy practices and preferences, WS-Federation for brokering and managing trust relationships (including identities) in a federated environment and WS-Authorization for authorization assertions and authorization policies.

The only one that is directly relevant to authorization is WS-Authorization which is planned to "describe how access policies for a Web service are specified and managed. In particular it will describe how claims may be specified within security tokens and how these claims will be interpreted at the endpoint", but which has not yet been published.

WS-SecurityPolicy defines policy assertions that have to do with security. It extends the securityToken to include SAML assertions and XrML licenses in addition to the identity values (x.509, KerberosTicket, UserName). It defines assertions about integrity, confidentiality and age of the message and of some properties of the security header. Since the other specs are not directly relevant to authorization they will not be discussed here.