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
Paul Madsen, Entrust
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 2003-09-23

GGF DOCUMENT SUBMISSION CHECKLIST (include as front page of submission)	
	COMPLETED (X) - Date
1. Author name(s), institution(s), and contact information	(X) – 2003-09-19
2. Date (original and, where applicable, latest revision date)	(X) – 2003-09-19
3. Title, table of contents, clearly numbered sections	(X) – 2003-09-19
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
Paul Madsen, Entrust
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 2003-09-23

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

Copyright © Global Grid Forum (2003). All Rights Reserved.

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. This document considers the various authorization concepts and their relationships and describes a framework that is expected to be general and abstract enough to allow for a classification of any type of authorization system. Abstract entities and functions are defined and fundamental communication sequences for authorization requests and decisions are shown. The mapping of existing authorization systems to the concepts introduced in this framework and an overview of related standards and protocols concludes the document.

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)

The checking of the proof of right can be done at a number of places:

- At the entrance of a service point (authorization may mean access control in this case).
- At a (central) point outside the service point.

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 high level entities involved in authorization. This terminology will be more refined during the course of the document.

Subject: Any entity (e. g. person or process) 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 Subject 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 may enforce access to these services based on a set of rules and policies defined by entities that are authoritative for the particular resource. Typical resources in Grid environments might be a computer providing compute cycles or data storage through a set of services it offers. Access to resources may be enforced by a Resource itself or by some entity (a policy enforcement point, gateway) that sits in front of a resource protecting it from being accessed in an unauthorized fashion.

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. These authorization policies contain assertions 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-pair. 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) defining an authorization policy at a high-level by a person or organization.
2) implementing the high level policy into a certain executable form
3) evaluating the executable policy by a process which subsequently decides to issue a specific authorization to a subject or take a specific action.

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.

The component performing the evaluation of the executable policy by computing an authorization decision on behalf of the authorities is sometimes referred to as an **Authorization Server**.

The term Authorization Server is however considered a rather vague term. Typically an authorization server may make or do (a combination of):

a. An Authorization Decision. Sometimes the term Authorization Decision Server is

used in this case. Authorization Decisions are typically the outcome of an evaluation of a policy,

b. An Authorization Lookup. A lookup of some entity's rights that are represented in some form and returned. These rights may form the basis of a new Authorization Decision taken elsewhere by another Authorization Server or they represent the outcome of a previously made (cached) Authorization Decision
c. The signing of a record of an Authorization as to assert its authority.
d. The delegation or proxy of an authorization decision to another Authorization Server.

One must therefore be careful with the handling of the term Authorization Server and be specific about a particular function of 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 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.

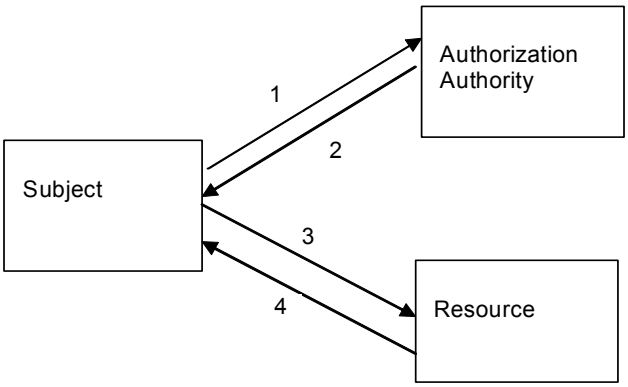

Figure 2.1 Authorization Push Sequence

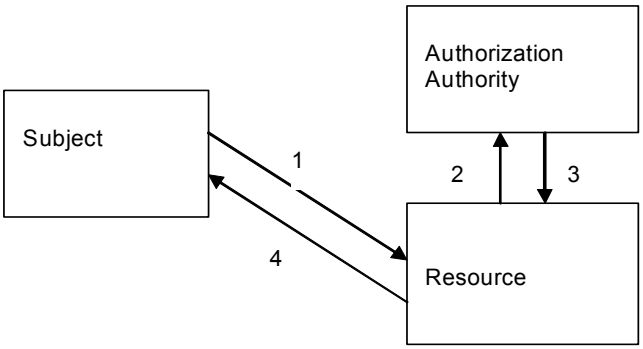

Figure 2.2 Authorization Pull Sequence

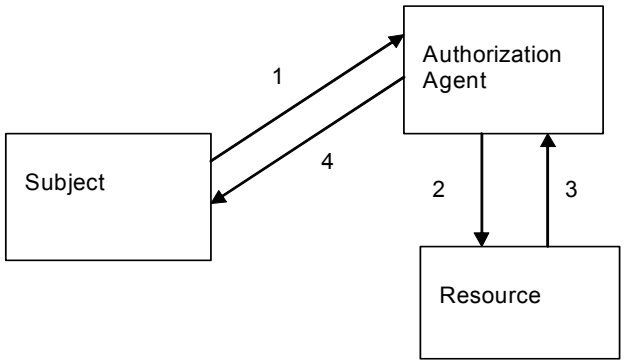

Figure 2.3 Authorization Agent Sequence

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 Resource. The Resource 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

An administrative domain is a definition of the scope of authority. In authorization scenarios there are at least two administrative domains: that of the Subject and that of the Resource. 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 Resource 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 typical Grid scenario is one where a Subject needs to use services from several domains. Sometimes this is accomplished by a Resource in one domain using a Resource in another domain on behalf of the Subject. Grid Service Providers may provide resources to users in multiple VO or home domains. For a more thorough discussion of multiple domain scenarios the reader is referred to appendix A which provides a taxonomy for the two-domain authorization scenario.

2.4 Hybrid authorization sequence 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). These hybrid sequences can be decomposed into assemblies of the three basic sequences.

2.5 Contractual and Trust relationships.

Contractual relationships between the domains of the different Subjects, Authorities and Resources 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 Authorization 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 authorization mechanisms, 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 issuing 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.

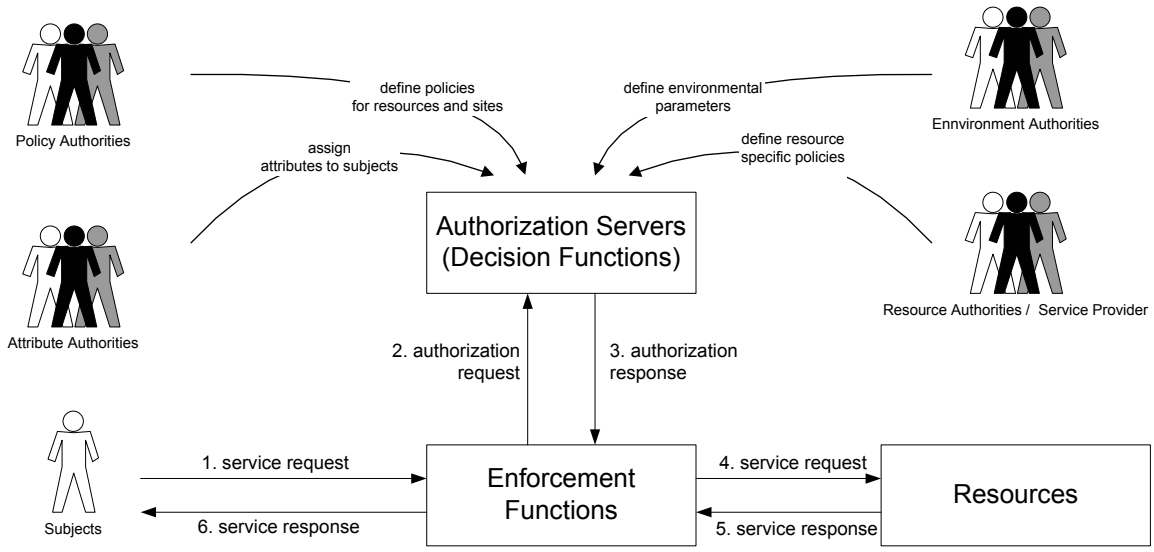

Fig. 3.1 Authorization Architecture based on the pull model.

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 most often either integrated into or located in front of Resource it protects.

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 Resource.

We additionally define two authority 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.

Authorization functions are embedded inside one or more administrative domains and may appear in a variety of combinations:

1. All authorization functions are combined in a single domain
2. The subject and resource functions are combined, the authority functions are independent
3. The subject and authority functions are combined, the resource functions are independent
4. The authority and resource functions are combined, the subject functions are independent
5. All entities are independent

Except for combination 1, contracts must govern the relationship and roles between the administrative domains. Specific interfaces or messages are needed to implement trusted and reliable communication of authorization decisions between administrative domains.

3.3 Flow of authorization information

Figure 3.1 simplifies the information flow in an authorization system based on the pull model. Attributes, parameters and policies, 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 authority 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.

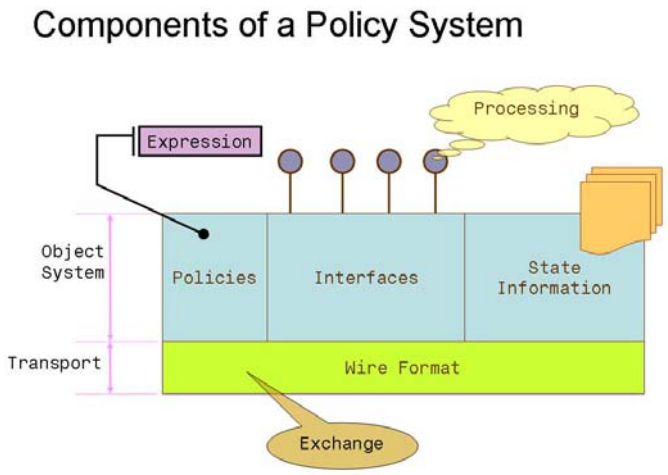

Figure 3-2 Components of a policy system – conceptual policy layers

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, exchange 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-on-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

The policy processing engines are likely to be proprietary to the various systems. Existing policy systems, including those based on AI and neural net paradigms, can be effectively used 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 systems 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.

4. Framework Components

4.1 Trust Management

4.1.1 Trust Authorities

Figure 3.1 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 3.1, 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 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 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 role based access control), 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.

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 Certification 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 3.1, 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 conditions (i.e. validity time) tied to a response, a priority level or a denial that includes information about the reason. The advantages of elaborate authorization responses have to be carefully weighed against their disadvantages. I.e. a condition specified in a response requires additional decision making logic in the enforcement mechanisms and possible reasons given for negative responses may allow an attacker to gather security data and reconstruct access policies.

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) in advance. Rather it is important to control 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 with the service request (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 and 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. For example CAS, Akenti, and 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 for example in the PRIMA system.
- Resource access by the application is intercepted and evaluated before passed on to the operating system. Sometimes referred to as "sandboxing", this approach 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 discuss a selection of existing authorization systems in this section in relation to the authorization framework mechanisms addressed in the earlier sections.

5.1 Akenti Authorization Service

5.1.1 Model and Architecture overview

Akenti provides an access control decision function and can be used in both a push or pull model. At the most fundamental level it takes the identity of the requester and the name of the resource and returns the access rights of that user in a signed capability certificate, aka an authorization assertion. In the pull model the user makes an authenticated connection to the resource, which then calls Akenti, passing along the user and resource name. In a push model, the requester calls Akenti though a possibly unsecured connection with its name and the name of the resource. Akenti returns a signed authorization assertion that contains the name of the requester, the name of the resource and the access rights. The requester passes that assertion though an authenticated connection to the resource, which checks to see that the name in the assertion matches the authenticated name and verifies the signature on the assertion. Akenti expects all principals to be identified by X509 public key certificates. Its ADF interface can be passed attribute assertions and desired rights to limit the scope of the attribute search.

5.1.2 Attribute Assertion and Policy Assertion functions

Akenti provides both attribute and policy assertion functions. It allows the stakeholders to create signed XML certificates (aka assertions) containing attribute assertions and policy assertions (aka policy certificates and use conditions). These can be stored in a distributed fashion and contain enough information for the Akenti ADF to find them. These assertions are stored and passed between the parties as signed XML formatted certificates that contain a unique certificate id and version, the signature algorithm, the certificate type, begin and end validity times and the identity of the issuer.

5.1.3 Flow of authorization information

The stakeholders store the certificates they have created in attribute and policy repositories that can be accessed with via http, ldap or file system requests. The communication between a requester and the Akenti ADF is via SOAP protocol carrying simple query and reply messages. An AEF can either link in the Akenti ADF libraries or communicate with an Akenti server via SOAP messages.

5.1.4 Trust Management

Akenti expects all trust relationships to be explicitly stated as part of the authorization policies. It expects all principals to be identified by X.509 public key certificates and uses the public keys to verify all policy and attribute assertions at the time of the access decision. The root policy for a resource domain contains the X.509 certificates and publishing directories for all the trusted CAs, and a list of the stakeholders for the domain. The stakeholders are the only entities that can issue use conditions for the resources. The use conditions state the required attributes and values needed to access the resource and who may issue attributes with those values. Whenever a use condition or attribute assertion is used, Akenti checks that it was issued by an acceptable party and verifies the signature to guarantee the integrity of the assertion.

5.1.5 Enforcement Mechanisms

Akenti is basically a decision function and leaves the enforcement to the resource management. It does supply an API to help the AEF evaluate run-time conditions. The authorization assertion returned by Akenti may include run-time conditions such as time limitations or load factor limits that the AEF needs to evaluate. Akenti does provide an evaluation framework API that can be used to combine such conditions if the AEF supplies functions to evaluate the individual conditions.

5.2 Cardea

Cardea [CARDEA] is a distributed authorization system, developed as part of the NASA Information Power Grid, which dynamically evaluates authorization requests according to a set of relevant characteristics of the resource and requester rather than considering specific local identities. Potentially accessed resources within an administrative domain are protected by local access control policies, specified with the XACML syntax, in terms of requester and resource characteristics. The information needed to complete an authorization decision is assessed and collected during the decision process itself. This information is assembled appropriately, either by the requester, an agent, a policy enforcement point (PEP), or a SAML policy decision point (PDP) and presented to an XACML PDP for evaluation. Once obtained, the SAML PDP then returns the final authorization decision for the access request together with any relevant details to the initial requester.

5.2.1 Authorization information

Any characteristic of the subject, the requested resource, the desired action or the current environment may be considered in the authorization decision. The model adopted by Cardea represents these attributes as SAML assertions that are passed between components. Each component is free to use the assertion data in any capacity it needs, such as transforming it to a different native internal format. However, when communicating the data between components, all characteristics are represented in this common format, regardless of the source or guarantor.

5.2.2 Initiating and enforcing the authorization decision

Cardea leverages the XACML model for authorization evaluation and SAML for obtaining assertion data used during the evaluation process. Cardea assumes that the SAML PDP that accepts the initial request is responsible for providing the final authorization decision details to the PEP. The SAML PDP depends upon the content of the initial request to determine the correct XACML PDP to evaluate the request. Then, the flow of communication between entities is specified by these relevant standards.

5.3 CAS

The Community Authorization Service (CAS) [CAS02, CAS03] allows for a separation of concerns between site policies and VO policies. Specifically, sites can delegate management of a subset of their policy space to the VO. CAS functions as a "push-model" authorization service, as shown in Figure 4. In this section we give a brief overview of how CAS is used in normal operation.

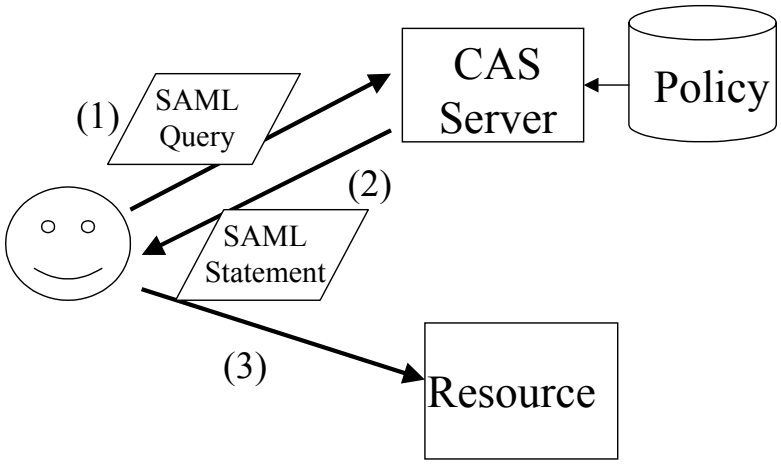

Figure 4: CAS Architecture. Steps are described in the text.

The steps in the Figure are:

1. The client, shown at left, sends a signed SAML AuthorizationDecisionQuery request to the CAS server, at right, indicating which resources they wish to access and which actions they desire to invoke.
2. The CAS server establishes the user's identity. Using the identity it determines the rights as established by the VO's policy. It then returns a signed SAML assertion containing an AuthorizationDecisionStatement. This assertion has the user's identity as the Subject and some subset of the user's requested actions
3. The user presents the SAML assertion to a resource along with an authenticated invocation request. The resource uses the SAML assertion, subject to local policy regarding how much authority was delegated to the CAS service, to authorize the request. The user may use the assertion to potentially make multiple requests, potentially to multiple resources.

Note that it is common for a client to ask for an assertion containing a complete set of rights they may have on a given resource, set of resources, or even on all resources for which a CAS server has authority. Since the SAML statement returned is typically valid for a number of hours, a assertion with multiple rights allows the user to undertake a number of different actions, which may not be known a priori, without having to re-contact the CAS server.

5.4 PRIMA

PRIMA [PRIMA02, PRIMA03] is a system for privilege management and access control. It provides tools for end users and administrators to manage privileges for the resources they are authoritative for through grid middleware mechanisms. PRIMA leverages X.509 Attribute Certificates to carry privilege and policy statements. An access control decision function authorizes requests based on the combination of subject attributes (privileges) with resource policies and provisions low-level access control enforcement functions with decision qualifications. These enforcement mechanisms assign and configure local user accounts dynamically and leverage POSIX file system access control lists and the XML based grid access control lists (GACLs) of the Slashgrid project to assign and manage fine grained access rights.

5.4.1 Authorization sequence

Prima uses a hybrid authorization sequence. When a subject requests a service from a resource, an enforcement function (AEF) queries an authorization decision function (ADF) for a coarse

access decision (yes/no) and a handle to an execution environment within which the service should be executed. This conforms to the flow outlined in the pull scenario. However, the ADF also configures an execution environment (enforcement function) for the service with the least amount of fine-grained rights required by the service. This second step in which decision qualifiers are provisioned to an AEF follows the information flow described by the push sequence.

5.4.2 Enforcement Mechanisms

Enforcement in PRIMA's enforcement is based on controlling the environment within which the application will execute. The PRIMA AEF creates, configures and manages local user accounts on-demand, based on subject privileges issued by authoritative entities such as resource administrators, resource owners, or group and project leaders. It utilizes dynamically modified file access control lists and host based firewall rules to constrain services. Enforcement via the execution environment enables PRIMA to securely execute non-trusted legacy applications without duplicating security code already present in the operating systems.

5.4.3 Decision Function

PRIMA's ADF makes access decisions based on subject attributes (privileges) and access control as well as privilege management policies. The subject can specify what attributes will be considered by the ADF by selectively providing them with the request. The PRIMA approach constitutes an additive, capability-based security model where missing or deliberately omitted attributes will result in fewer access permissions. The ADF is co-located with the resource and communicates with the gatekeeper's authorization module via XACML authorization requests and responses. Subject privileges are supplied to the ADF in the form of a dynamically created policy that is specific to the request and encompasses all the privileges that the subject presented.

5.4.4 Attribute Assertion and Policy Assertion Function

PRIMA provides tools for authoritative entities to create subject attributes that award individual privileges to a subject (holder). The privileges are encoded as XACML rule constructs and embedded in X.509 attribute certificates, signed by the issuer.

A tool for policy creation that aids administrators in creating resource access and privilege management policies in XACML is being developed. The policies are embedded in attribute certificates for secured transport to the PRIMA policy decision point where they will be enacted. Two types of policies are used, a resource access control policy is used to constrain or augment the privileges awarded to a subject via subject attributes and a privilege management policy defines who is authoritative for subject privileges and how such privileges may be delegated.

5.5 PERMIS Authorization Infrastructure

PERMIS is an attribute based authorization infrastructure comprising the following components

i) An authorization policy written in XML, digitally signed and secured as an X.509 attribute certificate
ii) User authorization tokens, which are attribute certificates conforming to the X.509 standard
iii) One of more LDAP directory servers that are used to store the attribute certificates (policy and authorization tokens)
iv) an ADF
v) API to the ADF, that is called by the application dependent AEF, and is very approximately a simplification of the OpenGroups AZN API.
vi) Assorted tools for creating attribute certificates and policies

5.5.1 Authorization Framework

PERMIS currently works in the authorization pull model. The user contacts the resource (or more precisely the AEF component of the resource), which in turn contacts the PERMIS ADF. The PERMIS ADF makes the decision, based on the user's attributes (obtained from the user's attribute certificates) and the policy for the resource, and returns this to the AEF. The AEF then enforces this decision on behalf of the resource. The PERMIS decision-making is two stages. The first stage, which typically takes place at user log on, is to evaluate the user's attribute certificates according to the policy, and to reject untrustworthy ones. Valid attributes are kept and returned to the AEF in a Java object. The second stage, which typically takes place when the user attempts to perform some action, is to make a grant or deny decision based on the user's validated attributes and the policy. Authorization requests and responses are passed as parameters in the Java API, and the decision is a simple Boolean.

5.5.2 Flow of Authorization Information

Attribute acquisition is normally done by a Privilege Allocator tool that creates X.509 attribute certificates and stores them in an LDAP server, in the entry of the holder of the attribute certificate. Attribute application can be via either the attribute push or pull model. In the attribute pull model, the PERMIS ADF is configured with the URLs of the LDAP servers it is to contact, and it retrieves all the X.509 ACs that it can find for the user making the access request. In the attribute push model, the AEF passes the attribute certificates that are to be used to the PERMIS ADF. How the AEF gets these is an application dependent issue. The user could send them to the resource along with his access request, or a Shibboleth type service could fetch them from a remote attribute server..

5.5.3 Policy Issues

The PERMIS policy is written in XML and comprises a set of sub policies. Full details of these sub policies can be obtained from papers on our web site (http://sec.isi.salford.ac.uk). The policy supports hierarchical RBAC, whereby users are given roles (or attributes) and roles/attributes are granted access rights. Superior roles/attributes inherit the privileges of subordinate roles/attributes. The policy can contain arbitrary condition statements, such as GT, LT, EQ, etc that can be ANDed and ORed together.

The policy is stored as a digitally signed attribute certificate in an LDAP directory by its creator, the Source of Authority (SoA) for a resource. The PERMIS ADF Java object retrieves the policy from this LDAP directory during construction time. No external processing agent sees the PERMIS policy, since the PERMIS ADF makes all decisions internally, and a simple Boolean grant/deny response is returned (i.e. no policy condition statements are returned to the AEF).

5.5.4 Trust Management

The PERMIS ADF trusts the resource, and the resource AEF provides PERMIS with the name of the authorization root of trust (the resource SoA name) when the PERMIS Java object is constructed by the AEF. The AEF also provides PERMIS with the unique OID of the policy to be used, and the URLs of the LDAP directories to be contacted. The PERMIS ADF then contacts the configured LDAP directory(ies) and retrieves the policy attribute certificate(s) from the entry of the resource SoA. PERMIS checks that the policy is signed by this SoA and has the correct OID, so that the policy can now be trusted. The policy contains the names of remote SOAs who are trusted to issue attribute certificates to users. In this way the resource SoA has delegated authority to a set of remote SoAs who are now trusted to issue attribute certificates to their users. Attribute certificates must be signed by one of the trusted SoAs and conform to the Role Assignment SubPolicy or they will be discarded by the PERMIS ADF. Dynamic delegation by remote SoAs to their subordinates is currently not supported. If a subordinate issues an attribute certificate to a user, it will not be trusted.

Attribute certificates contain the distinguished names of their holders (subjects). In PERMIS, the root of trust in authentication is the responsibility of the application, and PERMIS trusts the application to properly authenticate the users and to validate the digital signatures on attribute certificates. A user must authenticate himself to the application to prove that he is identified by a given DN, and then PERMIS can trust that attribute certificates containing this DN belong to the user.

5.6 The EU DataGrid Security Architecture

The EDG Security Architecture [EDG-SEC] is based on two types of authorization component: Virtual Organization Membership Services (VOMS) managing attributes, and several Authorization Decision Functions available to resources. VOMS servers respond both to authorization pull requests from resources, and also to attribute assertion requests from subjects wishing to use resources. Consequently, this architecture supports both the Push and Pull models described earlier.

5.6.1 VOMS Attribute Authorities

In the Pull model, the VOMS server is periodically contacted by each resource using HTTPS, and requests are made for listings of members' certificate subject names matching specified criteria (for example, all the members of a given group.) Since the VOMS HTTPS server identifies itself by its own certificate at the start of the connection, no additional signing of this information is used.

In the Push model, subjects wishing to use resources contact the VOMS server. Client tools at the subject's home location identify the subject to the VOMS using a GSI proxy, and request the desired set of attribute assertions for the subsequent session. The VOMS server issues a signed text block of name-value pairs containing those requested attributes that the subject is entitled to. The text block starts with the subject and VOMS certificate names, the name of the VO, and upper and lower time limits on the validity of the assertion. One or more Group, Role, Capability triplets then follow, with value NULL if no specific Role or Capability is being asserted. This signed assertion is then included as an extension in a new GSI proxy for the subject, generated by the subject's client software.

5.6.2 Authorization Decision Functions

The Authorization Decision Function is provided via libraries which applications link to via the application's native Authorization Enforcement Function. Three complementary systems are provided: GACL, LCAS and the Java Authorization Manager.
GACL is a library for processing Grid Access Control Language policy statements, written in XML. These statements grant permissions (such as write) to subjects satisfying one or more criteria (such as a specific subject name or membership of a VOMS group.) The GACL C/C++ API provides an Authorization Decision Function, returning yes/no answers given a proposed action (eg write), the credentials the subject possesses and the policy associated with the object in question. The Local Centre Authorization Service (LCAS) provides a more flexible and customizable framework for Authorization Decision Functions, in a way suitable for heavier weight requests, such as transfers of large files and job submissions. Decisions themselves are carried out by plug-ins to the framework. Plug-ins exist to use a list of subject names (such as a Globus grid-mapfile) or list of VOMS attributes, or to use a policy statement written in GACL. The Java Authorization Management system provides an Authorization Decision Function for Java-based SOAP services. XML policies grant internal attributes to subjects on the basis of certificate name or VOMS attributes.

6. Related Standards

Various groups at GGF, IETF and OASIS are involved with the standardization of various elements of authorization frameworks. In this section we try to give a summary of these related activities and how it is related to the areas covered in this document.

Security Assertion Markup Language (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 a variety of systems. SAML provides schemas for queries and replies and for security related assertions. The assertion statement is one of: authentication statement, attribute statement or authorization statement.

eXtensible Access Control Markup Language (XACML) is an XML Specification under development for expressing policies for information access over the internet.
The schema defines the elements needed to describe an authorization policy and to describe the context in which a request for authorization is made. 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. The rules are combined using a rule-combining algorithm into policies which include obligations that must be met when the actions are performed.

eXtensible rights Markup Language (XRML) is a language to describe the rights and conditions for using digital resources.[XRML] 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.

Web Services Security is an attempt within the Web Services community to provide a standard XML vocabulary for defining the entire range of security issues in distributed systems. Web services communicate via SOAP messages and 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, in a transport-neutral way. The WS-security framework defines seven specifications of security functionality. The one that is directly relevant to this discussion 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 defines assertions about integrity, confidentiality and age of the message and of some properties of the security header.

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.

RFC2704 The KeyNote trust management system [RFC2704] 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.

RFC2904 – The AAA Authorization Framework [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. A PEP is the corresponding entity, typically located at the service equipment, that can enforce an access decision made by the PDP.

The ISO 10181-3 [ISO 10181-3] authorization framework defines four roles for components participating in an access request. They are Initiators, Targets, Access Control Enforcement Functions (AEF) and Access Control Decision Functions (ADF). An Internet Attribute Certificate Profile for Authorization (RFC 3281) [RFC3281] provides a means to bind arbitrary attributes (e.g. role membership information, policy statements, accounting information) to identities. Authorization API – Generic Application Interface for Authorization Frameworks defines a standardized API for the interface between AEF and ADF as defined in RFC2904. The IETF proposed standard Generic Authorization and Access control API (GAA API) 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.

7. Security Considerations

The discussion of security concerns is intrinsic to this document.

<Bob: Comment for discussion:
After reading our document, I'm concerned about the huge potential for information leakage in this authorization system. Either in the direction of User -> resource as the user supplies all of their attributes in an attempt to cajole the resource authorization server to grant access permission without additional hassle ... or in the direction of Resource -> User if the user can query what permissions are needed to get a request granted.

Anyway, it feels like something that is missing is a risk analysis ... or some thinking about how these structures could be easily attacked and where we need to be especially careful. It's something to cary over into the OGSA work, maybe.>

Mary's suggestion:
When informing the user that access has been denied the
> implementer of an ADF needs to decide how much information about the
> reason for failure should be included. Returning no
> information is the
> most secure choice, but makes the system very difficult for
> legitimate
> users who need to know why an access failed in order to fix
> the problem."

Also see last sentence in 4.5
>

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

Paul Madsen
Entrust
1000 Innovation Drive,
Ottawa, K2K-3E6
Canada
Email: p.madsen@entrust.com

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
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

Acknowledgements

We would like to acknowledge the contributions from members of the community, specifically Carlisle Adams, Von Welch and David Chadwick.

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

Copyright (C) Global Grid Forum (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the GGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

References

[Akenti]
M. Thompson, A. Essiari, S. Mudumbai, “Certificate-based Authorization Policy in a PKI Environment,” *ACM Transactions on Information and System Security*, Aug 2003

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

[CARDEA]
<!--RSL writes:
As per Cardea documentation, there should be a NASA technical report published in the near future that addresses the system. There will also be a poster/demo at SC2003 in November and a WiP at ACSAC2003 in December. Materials should be forthcoming on the IPG website for each of those.-->

[CASR2]
CAS AlphaR2 Web site, http://www.globus.org/Security/cas/alpha-r2/ , September 2002.

[CAS02]
Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[CAS03]
Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., The Community Authorization Service: Status and Futures. Computing in High Energy Physics (CHEP03), 2003.

[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/

[EDG-SEC]
R. Alfieri et al. (EDG Security Co-ordination Group), "Managing Dynamic User Communities in a Grid of Autonomous Resources", Proceedings of Computing in High Energy and Nuclear Physics (2003).

[GAA-API]
http://www.isi.edu/gost/info/gaaapi/

[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.

[PRIMA02]
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

[PRIMA03]

Markus Lorch, David Adams, Dennis Kafura, Madhu Koneni, Anand Rathi, Sumit Shah, "The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments", accepted for publication at the 4th Int. Workshop on Grid Computing - Grid 2003, 17 November 2003 in Phoenix, AR, USA

[RFC2704]
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

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

[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

[Shibboleth]
Internet2 Shibboleth Project, “Shibboleth Architecture “ http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-arch-v05.pdf , 2 May 2002

[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

[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: Two-domain authorization models taxonomy

Two-domain authorization
A domain is a set of entities (subjects and/or resources and/or other) whose membership is controlled according to a policy that is administered by an authority.

Two-domain authorization models arise when an authority in one domain administers the set of resources for which authorization is required, but one or more components in the authorization architecture reside in another domain. This situation is often encountered in Business-to-Business as well as grid computing scenarios, especially when resources from a number of physical organizations are pooled to form virtual organizations

Two-domain authorization models taxonomy
We consider the different combinations by which the different components of an authorization framework (Attribute Assertion Function (AAF), Access Control Decision Function (ADF), Policy Assertion Function (PAF), Access Control Enforcement Function (AEF), Resource Assertion Function (RAF)) can be distributed across a domain boundary (with the constraints that the AEF is located in the same domain as the resources it protects, and that the Resource Authority is located in the same domain as the resources for which it is the authority). With two of the five components fixed, we can imagine the different combinations in which the remaining three components (AAF, PAF, and ADF) are moved to an external domain. Thus, there are 7 combinations of which components can be moved to the external domain (see Fig. A-1); labeled as Remote AAF, Remote PAF, Remote ADF, Remote AAF & PAF, Remote AAF & ADF, Remote PAF & ADF, and Remote AAF & PAF & ADF.

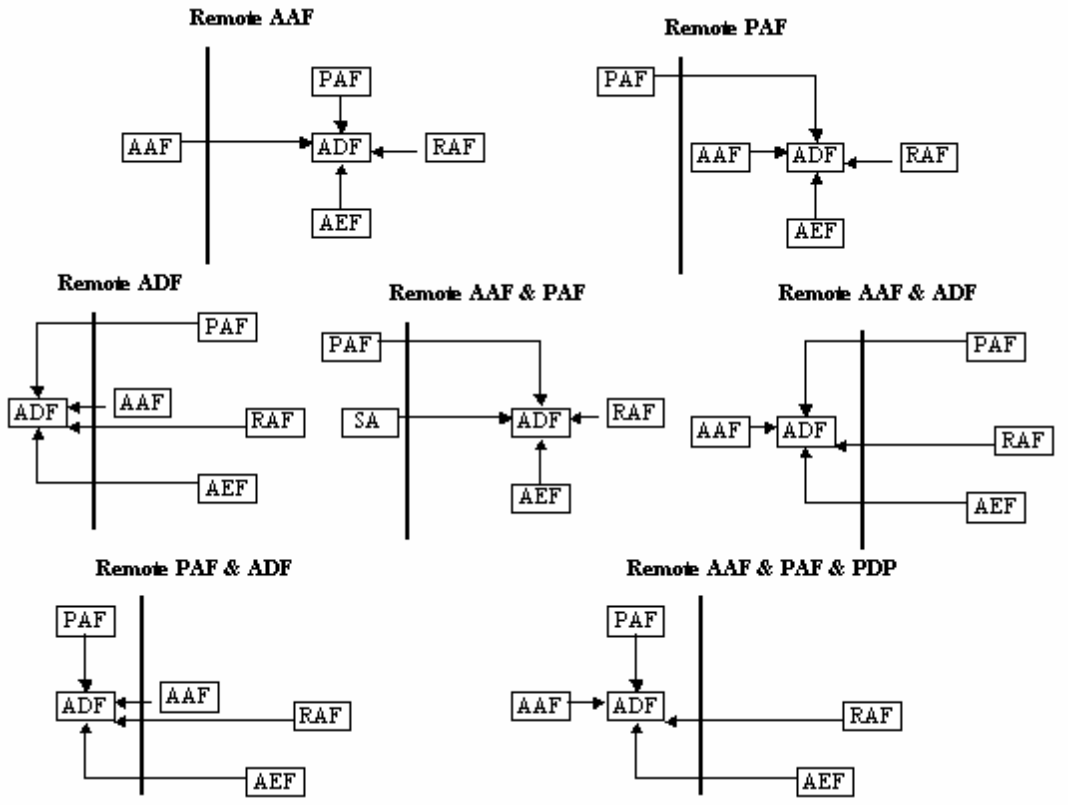

Fig. A-1: Two-domain authorization models

It's important to note that the above diagrams are intended to show configurations of components, not information flows. For instance, for any configuration that involves the sharing of subject attributes, these may be included in the actual request to access a resource, or they may be requested by the AEF (or other component) from the subject after receipt of the access request, or they may be requested by the AEF (or other component) from an Attribute Authority in the subject domain, or they may be retrieved from some form of online repository (such as LDAP or a wallet). Similar options are available for retrieving the resource attributes, the environmental attributes, and the policy upon which to base the decision. As well, the order in which this data is retrieved may vary from environment to environment, or even from request to request. Consequently, a variety of information flows may be implemented for any given configuration of components in the two-domain authorization architecture.

We explore briefly the advantages and issues associated with each of the 7 configurations in the sections below.

Remote AAF
In this configuration, a Subject Authority in the Subject's domain facilitates the subject's access to resources in other domains by making available relevant attributes of the subject to these resource domain. Subject attributes, such as role, signing authority, security clearance, etc., are administered in the subject domain and communicated to the resource domain. This model is defined in standards such as X.509 [RFC3281], SAML [SAML], and Shibboleth [Shibboleth].

There are three obstacles to the success of this approach. The first is the need for the subject domain and all the resource domains to agree upon the syntax and semantics of the subject attribute encoding. The second is the need to agree upon a mechanism for communicating the attributes. The third is the privacy concern inherent in the exchange of potentially sensitive attribute information across the network.

There has as yet been only limited success in cross-domain agreement (in national or international standards bodies, or in industry verticals) on syntax and semantics for subject attributes.

Remote PAF
In this configuration, an authority from another domain creates a policy (perhaps *the* policy) upon which the ADF acts. Such a configuration is well suited to the large corporate environment in which corporate policies created by the head office may dictate/influence authorization decisions made with respect to local resources in a branch office. However, this model is also appropriate for cases in which legislative or regulatory policies are imposed upon local environments within a particular vertical industry segment. An example of this may be the U.S. Health and Human Services department defining a policy based on the U.S. Health Insurance Portability and Accountability Act (HIPAA) which must be adhered to by individual healthcare organizations. *Note:* This model actually implies that there are 3 domains relavent to the authorization decision; that in which the policy is defined, that in which the resource is held, and that in which the subject is a member.

Remote ADF
In this configuration, the ADF is outsourced – perhaps implemented as a Web service – to another domain. The resource domain sends all relevant information (policy, attributes, access request, and so on) to this ADF and receives an authorization decision in return. Although it may seem that the resource domain is doing all the difficult work (understanding the policy well enough to know what all the relevant inputs are; finding all those inputs; and validating each of them for authenticity, integrity, and freshness) and outsourcing just the simplest task (processing the policy and inputs to arrive at a decision), there may conceivably be some merit in this model. The outsourced ADF may be a Decision Authority recognized in many domains, so that its decisions have legal or similar value (e.g., for audit/archival purposes, to off-load risk for a faulty authorization decision).

Remote AAF & PAF
In this configuration, both the subject attributes and the policy come from a domain external to the protected resources. This model is appropriate to Digital Rights Management (DRM) scenarios in which the subject arrives with its own attributes and some form of policy (perhaps a "ticket") supporting its request to access a resource. The model also fits environments addressing privacy, in which the subject submits its own privacy policy, or personal preferences, that must be taken into account by the resource-domain ADF.

Remote AAF & ADF
In this configuration, a ADF in the subject domain makes authorization decisions in accordance with a policy created in the resource domain. This model ("here is the request, the resource, and our policy; tell us if your subject should be allowed access") seems unlikely to be commonly used. One possible application is for a type of pre-check: the subject checks with its own ADF to see what requests might be allowed before sending an actual request (and its subject attributes) to the resource domain. This pre-check can perhaps save some bandwidth and preserve privacy because sensitive attributes are not sent to the resource domain until the subject is relatively confident that the attributes are necessary and sufficient for the desired access.

Remote PAF & ADF
In this configuration, an outsourced ADF operates under its own policy to make authorization decisions with respect to subjects that are known and administered in the resource domain. This model may apply to cases in which the policy is legislative or regulatory and the ADF acts as an audit or compliance test center.

Remote AAF & PAF & ADF
In this configuration, the ADF operates in the subject domain according to its own policy. The ADF issues authorization decision assertions upon which the AEF in the resource domain can act. As shown in the figure, this is an over-simplification as the resource domain would almost certainly have its own ADF acting in accordance with its own policy.

This model, as it has the AAF and RAF in separate domains, is similar to the Remote AAF configuration, and like that configuration, seems an intuitive fit for a 'simple' bilateral B2B relationship (with no other 3rd party involved). Unlike the Remote AAF configuration however, this model avoids the privacy concern associated with subject attributes crossing the domain boundary by instead bringing the ADF and PAF 'across'-obviating the need for subject attributes to 'travel'.

Information Sharing
The different configurations presented in the previous sections are shown in the table below - identified by the types of information that must be shared across the domain boundary for each.

	Subject Attributes	**Resource Attributes**	**AuthZ Decision**	**Policy**
Remote AAF	✓			
Remote PAF	*	*		✓
Remote ADF	✓	✓	✓	✓
Remote AAF&PAF	✓	*		✓
Remote AAF&ADF	*	✓	✓	✓
Remote PAF&ADF	✓	✓	✓	
Remote AAF&PAF&ADF		✓	✓	

For Remote PAF, Remote AAF & PAF, and Remote AAF & ADF, the '*' symbol indicates that, although no *specific* attributes must be shared across the domain boundary for these

configurations, definition of a relevant policy may be dependent on the attribute categorization. For instance, a Remote PAF will need to know that Subjects are either Employees or BoardMembers in order to define a relevant authorization policy even though it would not be necessary in these cases for an assertion of the form 'Joe is a BoardMember' to cross the boundary. This requirement is characteristic of configurations in which the PAF is separated from either the AAF or RAF.

We note from the table that all of Remote ADF, Remote AAF & PAF, and Remote PAF & ADF, because they all involve Subject Attribute Sharing, will face the same privacy concern that was identified for Remote AAF.

We can also see that Remote PAF, Remote ADF, Remote AAF & PAF, and Remote AAF & ADF, because they all involve a separation between the PAF and the ADF, all require that policy must cross the domain boundary (for Remote PAF in the opposite direction from the others). At the moment, there are very few standard protocols and data structures to carry such information. The X.509 attribute certificate work did specify a way to carry a PrivilegePolicy construct in an attribute certificate, but this has not been widely adopted or deployed. There has also been some discussion of extending the SAML "Statement" type to define a "PolicyStatement" that can then be carried in a SAML assertion along with other kinds of statements, but this work has yet to begin.

We have not attempted to account for environment attributes in our discussion. The deciding ADF may require this type of input (e.g., time of day, current stock price, etc). This data may come from the domain in which the ADF is operating, or may need to cross the domain boundary along with some of the other inputs. This level of detail is not shown in the above diagrams because the source of environmental data necessarily depends upon the specific request made and the particular policy that governs the access decision; it is not possible to say in a definitive way that environmental data will always come from *this* domain or *that* domain in a given model.