Authorization and Policy

Overview and Definitions

Authorization for resources served by the services running on the Bamboo Services Platform (BSP), or resources for which the BSP manages policy, is architected to the XACML 2.0 specification (PDF). XACML is an OASIS specification; the acronym stands for *eXtensible Access Control Markup Language*. Some elements of the XACML spec were implemented in a manner that offers less-than-ideal usability in phase one of the Bamboo Technology Project, e.g., the Policy Access Point is implemented as a directory into which a system administer writes policy files prior to activation of the Policy bundles (see *Deploying Policies*, below), instead of (for example) a database with an appropriate service API and user interface for easier, permission-controlled administrative access.

Authorization decisions are based on a policy-driven process that takes into account:

- rules previously established for the resource in question;
- the action requested;
- a set of user data that includes identity, their attributes, and any group memberships; and,
- environmental factors, such as date and/or time.

It is important to note that "the resource in question" may be a resource provisioned by a service, or the service itself. In the latter case, a policy is governing rights to make requests of a service.

The process of authorizing a client's request can be broken down into four steps:

1. Intercepting the request and harvesting passed attributes
2. Resolving the attributes required to make a decision
3. Deciding whether to allow or deny the request
4. Logging the request

Here are some useful concepts and definitions:

Policy Enforcement Point (PEP): The system entity that performs access control by making decision requests and enforcing authorization decisions. See *Policy Enforcement*, below.

Policy Decision Point (PDP): A service that assembles the needed information to evaluate a policy with respect to a concrete request for which authorization is required, evaluates the policy, and boils the answer down to ALLOW or DENY (plus, optionally, some additional obligations). In Bamboo's implementation, the PDP is accessed through the *Request Manager service*.

Policy: A policy is a set of rules; a rule consists of a target, an effect, and a condition. See *Policy Examples*, below. Policies are XML documents that conform to the XACML schema for Policy Sets.

Policy Information Point (PIP): a system entity that acts as a source of attribute values. See *Factors used in deciding Policy*, below.

Context Handler: The system entity that converts decision requests in the native request format to the XACML canonical form and converts authorization decisions in the XACML canonical form to the native response format. In Bamboo's implementation, this function is handled by the *Request Manager service*.

Subject / Principal: An actor whose attributes may be referenced by a predicate. This is a general way of describing a *user* whose attributes that matter vis-a-vis policy decisions are the *Factors used in deciding Policy*, see below.

Environment: The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action. An example of an environmental attribute that could figure into a policy is *time* (e.g., a policy might contain rules that boil down to "*this resource is only available on weekdays*").

Authorization Request: The roles, resource, action and environment data to use when making a policy decision. In Bamboo's implementation, the *Request Manager service* assembles these data and routes the authorization request to the Policy Decision Point.

Authorization Decision: The result of evaluating applicable policy, returned by the PDP to the PEP. A function that evaluates to “Permit”, “Deny”, “Indeterminate” or “NotApplicable", and (optionally) a set of obligations. In Bamboo's implementation, only "Permit" decisions are authorized; any other resolution is denied access to the requested resource.

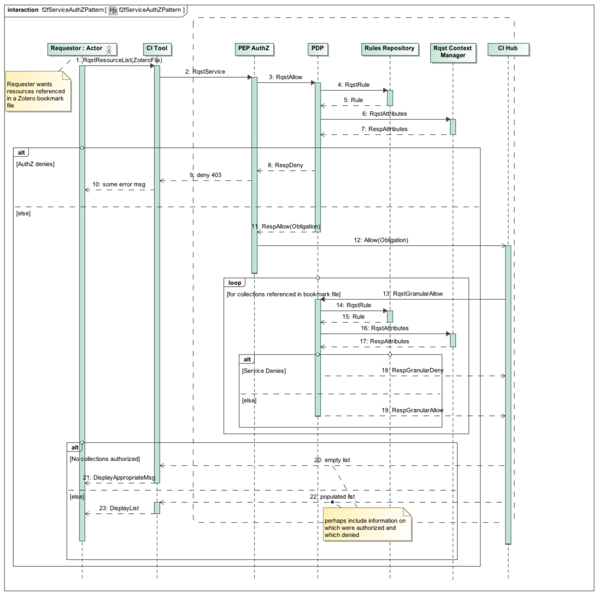
A comprehensive treatment of these terms and concepts can be found in the XACML 2.0 specification (PDF).

Policy Enforcement

Policy may be enforced at two conceptual points:

1. when a request arrives at the BSP, and is evaluated 'at the perimeter' as to whether or not it may be forwarded to a service hosted on the BSP; and/or,
2. when a request is being processed by a service, and the service requires an authorization decision based on policies that were not or could not be evaluated 'at the perimeter.'

The sequence diagram included here (click to enlarge) served as a model use case to which Bamboo's IAM infrastructure was built. Further explanation follows below.



Of particular note in the diagram:

- The *CI Tool* is the software used by someone to access the services provided by the CI Hub, a service hosted by the BSP that fetches and aggregates digital content from multiple repositories; repositories from which content is requested can have access restrictions that apply to them. The CI Tool might be a browser used to inspect content fetched by the CI Hub, and would be implemented as a client tool, external to the BSP.
- The *PEP AuthZ* component is a Policy Enforcement Point that assures the initiator of the request is allowed to access the CI Hub before passing the request through. This Authorization component governs access at *a service level* and has no knowledge of structure or content of the target-service's input parameters or input payload.
- The CI Hub is an example of a service that requires authorization at a *finer-grained level* than can be provided by the *PEP AuthZ*. This is because it functions by allowing a user to pass in a file containing a collection of requests for content from different repositories. Within the BSP, only the CI Hub knows how to parse the contents of this file.
    - For this reason, the CI Hub service acts as a PEP (Policy Enforcement Point) itself, making as many calls to the PDP (Policy Decision Point) as are necessary to obtain the authorization(s) it needs to fulfill a user's request.
        - The CI Hub is responsible for "unpacking" the input file (a Zotero bookmark file in this use case), inspecting it, and calling the PDP to determine whether the requesting user is permitted to access content from the repositories identified in bookmark URLs.
        - The CI Hub only checks policy for those URLs that it can act upon -- that is, those URLs that request content from a repository for which the CI Hub is capable of mediating access.
    - These multiple calls from the CI Hub to the PDP are represented inside the *loop* box in the sequence diagram
    - Not all services will require fine-grained authorization. Only services that do have this need are responsible for implementing calls to the PDP and enforcing decisions to ALLOW or DENY access, conforming to decisions returned by the PDP.

In phase one of the Bamboo Technology Project, no service implemented the 'inner loop' of this sequence diagram. That is, the only policies written, tested, and enforced at the time the BTP ended were those that govern access at a *service level*, as described in the second bullet above. However, the *Request Manager service* provides the interface a BSP-deployed service would need to obtain ALLOW/DENY decisions at the *finer-grained level* described in the third bullet above and its sub-bullets.

Policy Decisions

Policy decisions are accessed through the *Request Manager service*. They may be accessed by a service local to the BSP, or by an external client. In either case, the resources on which policy decisions are requested must be governed by a policy in effect (known to) the BSP AuthZ infrastructure (see *Policy examples...* and *Deploying Policies*, below).

Note that a resource may be brought under the governance of policy restricting/opening access to members of a Group or users holding a scoped role by associating the resource with a Group or scoped role using the *Protected Resource Service*. The owner (e.g., creator) of a resource is permitted to create such an association.

See the *Request Manager service API*, which includes links to the service codebase for those who wish to dive deeper into the implementation of the AuthZ infrastructure. External clients use the ROA Layer API; services deployed on the BSP use the SOA Layer API.

Factors used in deciding Policy

Supported types of policy decisions are given in the examples described in the following sub-section of this page. These decisions imply the ability of BSP to assemble a set of facts – including user attributes, registered application attributes, user group memberships, user scoped roles, and environmental conditions such as system time – that are checked against a policy for a given resource, resulting in an ALLOW or DENY decision vis-a-vis access to the resource.

The factors assembled for use in evaluating access include:

- **Institutional Affiliation** - In the current implementation, affiliation is inferred from an assertion passed to the Bamboo Services Platform by a trusted client application that authenticated the user; authentication by an institution (e.g., UC Berkeley or UW Madision) *in the current application session* is assumed to imply affiliation with that institution. The assertion passed by the trusted, authenticating client application is passed as a *sco ped role* in an HTTP header named *X-Bamboo-Roles*. See the *Client Responsibilities...* section of any service contract listed on the page *Service APIs - Centrally-Hosted Bamboo Services* for more information.
- **Scoped roles** - This is the general case, of which Institutional Affiliation (above) is a particular instance, of role assertions passed in the form *role @domain* (e.g., *user@clientapp.acmeuniversity.edu*) by the trusted, authenticating client application in an HTTP header named *X-Bamboo-Roles*. The owner of a resource may make it accessible to users holding a scoped role by associating the resource with a scoped role via the *Protected Resource Service*. See the *Client Responsibilities...* section of any service contract listed on the page *Service APIs - Centrally-Hosted Bamboo Services* for more information.
- **Group Membership** - Bamboo Persons can create and associate themselves and other Bamboo Persons with *groups*; see *Group Service Contract Description - v0.9-alpha*. The creator of a resource may associate it with such a group, so that members of the group are permitted access to the resource; see *Protected Resource Service Contract Description - v0.9-alpha*.
- **Resource Ownership** - A person who created a resource is considered its owner. A clear example of an owned resource is the Profile owned by the Bamboo Person who the profile describes. See API documentation: *Person Profile Service Contract Description - v0.9-alpha* and *Person Service Contract Description v0.9-alpha*. Also see *Group Membership*, above; the owner of a resource may make it accessible to members of a Group by associating the resource with a Group via the *Protected Resource Service*.
- **Request is sent by a trusted (or untrusted) client application** - A trusted client is one that is registered in the Bamboo Trust Federation, as described in the section *Bamboo IAM from a client application's perspective* on the page *Identity and Access Management - Authentication and Authorization*. An untrusted client is one that is not so registered. The exchange of certificates and metadata required to physically establish such a trust relationship is described in the *Certificate Exchange* section of the page *Configure Apache Web Server for Client Auth*. Limitations to the trust relationships enabled in the implementation built during the Bamboo Technology Project, the reason those limitations were encountered, and an expected path forward as authentication technology and technology adoption matures, is described in the page *Authentication - Current Limitations and Future Direction*.
- A **Bamboo Person registered for specific use by** a special-case **Innovation Licensed application** is making the request - An "Innovation Licensed application" is one that is trusted (registered, as described in the prior bullet); and is placed in a special category of application that is not required or expected to actually authenticate users, but is permitted nonetheless to assert the identity of certain, pre-identified Bamboo Persons (this could be thought of as permitted 'impersonation' of the pre-identified users). This purpose of this arrangement is to permit applications that are not equipped to participate in authentication as required by the Bamboo Trust Federation to nonetheless make authorizable requests. It is strongly recommended in such cases that the Bamboo Person identifiers associated with (and assertable by) an Innovation Licensed application be distinct from the identifiers used by any "real" person to authenticate (log in). See *Maintaining Application Catalog Data for Trusted Clients* for more on Innovation Licensed applications.

Supported policy decision examples

The following are examples that utilize factors described in the prior sub-section of this page. See the prior section for links to explanatory documentation and relevant APIs.

1. Access to a protected resource **requires affiliation with an institution** whose affiliates are permitted access to the resource. If the requesting user is affiliated with 'permitted' institutions s/he is allowed access to the resource.
2. Access to a protected resource **requires membership in a Bamboo-managed group** that has been granted access to the resource. If the requesting user is a current member of a group that has been granted access to the resource, s/he is allowed access.
3. Access to protected resource **requires both affiliation with an institution and membership in a Bamboo-managed group** (i.e. both of the above must be true).
4. Access to protected resource is ***denied* if the client application through which a request is made is *NOT* registered as an app in the Bamboo Trust Federation**.
5. Access to protected resource **is permitted if the association of individual (bpid) and requesting innovation app** has been registered, where the request is made by the Innovation Licensed application on behalf of a user it is allowed to 'impersonate.'
6. Access to protected resource **requires that the requesting user own the resource**, where "ownership" is a concept encapsulated by the service that models the resource. [*Example: a Person Profile may be UPDATEd only by the user whose BPId the profile describes; or by users in a group that has been specified, by the Profile owner, as a set of Bamboo Persons permitted to update the profile*]

User attributes as referenced in Policies

Attributes are the "factors" described above, in *Factors used in deciding Policy*. These are expressed formally in policies either as standard XACML attributes, or as attributes defined by Project Bamboo. Here is a list of attributes found in Policy examples referenced below, and what they represent.

These attributes are defined in two schema documents located in the Bamboo code repository at these locations:

✓ ${**REPOSITORY_ROOT**} is http://svn.code.sf.net/p/projectbamboo/code/

- ${REPOSITORY_ROOT}/platform-services/bsp/trunk/bsp/utility-services/protected-resource-service/protected-resource-service-domain/src/main/resources/ProtectedResource.xsd
- ${REPOSITORY_ROOT}/platform-services/bsp/trunk/bsp/core-services/request-manager-service/request-manager-service-domain/src/main/resources/Request.xsd

Attribute	Description
urn:mace:projectbamboo.org:attribute:1.0:app-id	Represents the Application Identifier that would be passed by the application making the request (cf. *Maintaining Application Catalog Data for Trusted Clients*). This attribute is passed by trusted client applications in an HTTP request header called *X-Bamboo-AppID*; see the *Client Responsibilities...* section of any service contract listed on the page *Service APIs - Centrally-Hosted Bamboo Services*.
urn:oasis:names:tc:xacml:1.0:subject:subject-id	Represents the Subject Identifier that would be passed by the application making the request (subject identifiers are passed by trusted client applications in an HTTP request header called *X-Bamboo-BPID*; see the *Client Responsibilities...* section of any service contract listed on the page *Service APIs - Centrally-Hosted Bamboo Services*)
urn:oasis:names:tc:xacml:1.0:resource:resource-id	Represents the resource that is the object of the request
urn:oasis:names:tc:xacml:1.0:action:action-id	RESTful actions that can be performed as part of a request (a.k.a. "HTTP verbs"). The allowed actions are: GET, PUT, POST, and DELETE.
urn:mace:projectbamboo.org:attribute:1.0:group-has-view-access-rights	Represents a group that is associated (via the *Protected Resource service*) with a protected resource, such that users who are members of the group have ***view*** access rights to the resource (cf. *Group Service Contract Description - v0.9-alpha*)
urn:mace:projectbamboo.org:attribute:1.0:group-has-update-access-rights	Represents a group that is associated (via the *Protected Resource service*) with a protected resource, such that users who are members of the group have ***update*** access rights to the resource (cf. *Group Service Contract Description - v0.9-alpha*).
urn:mace:dir:attribute-def:isMemberOf	Represents the Identifier of a group to which the subject (user) belongs (cf. *Group Service Contract Description - v0.9-alpha*).
urn:mace:projectbamboo.org:attribute:1.0:scopedRole	Represents an assertion that the user has some role within a domain. "Scoped roles" are passed by trusted client applications in an HTTP request header called *X-Bamboo-Roles*; see the *Client Responsibilities...* section of any service contract listed on the page *Service APIs - Centrally-Hosted Bamboo Services* . A "scoped role" is formatted as an RFC822 Name consisting of a local-part followed by "@" followed by a domain-part, e.g., *unspecified@berkeley.edu*.
urn:mace:projectbamboo.org:attribute:1.0:scoped-role-has-view-access-rights	Represents a "scoped role" that is associated (via the *Protected Resource service*) with a protected resource, such that users who hold the scoped role has ***view*** access rights to the resource. Cf. description of *urn:mace:projectbamboo.org:attribute:1.0:scopedRole* for additional context and formatting information.
urn:mace:projectbamboo.org:attribute:1.0:scoped-role-has-update-access-rights	Represents a "scoped role" that is associated (via the *Protected Resource service*) with a protected resource, such that users who hold the scoped role has ***update*** access rights to the resource. Cf. description of *urn:mace:projectbamboo.org:attribute:1.0:scopedRole* for additional context and formatting information.
urn:oasis:names:tc:xacml:1.0:subject:request-time	Indicates the time at which the subject initiated the access request.
urn:mace:projectbamboo.org:attribute:1.0:host-name	Represents the name of the server that will provision the request

urn:oasis:names:tc:xacml:1.0:environment:current-dateTime	Represents the time at the server that will provision the request
urn:mace:projectbamboo.org:attribute:1.0:app-contract-type	Represents the Contract Type that would be passed by the application making the request, where "contract type" refers to contractual relationships in the Bamboo Trust Federation, such as "Trusted Application" or "Innovation Licensed Application." Cf. *Maintaining Application Catalog Data for Trusted Clients*.

Policy examples: tested and versioned in Bamboo Code Repository

Policies are XML documents that conform to the XACML schema for Policy Sets.

Policy examples, which have been tested in the v0.9 release of the Bamboo Services Platform, are versioned in the code repository at multiple locations:

✓ ${**REPOSITORY_ROOT**} is http://svn.code.sf.net/p/projectbamboo/code/

1. Policies that pertain to BSP services are versioned in the BSP codebase as follows:
   a. Policies that pertain to a secured BSP: ${REPOSITORY_ROOT}/platform-services/bsp/trunk/bsp/utility-services/policy-service/policy-service-domain/src/main/resources/policies/
   b. Policy to ALLOW all requests, appropriate for unsecured instances (e.g., developer environment): ${REPOSITORY_ROOT}/platform-services/bsp/trunk/bsp/utility-services/policy-service/policy-service-domain/src/test/resources/policies/
   c. Note that a test policy for scoped roles is also included in the repository directory indicated in (b) above. *This policy (ScopedRoleTest.xml) is referenced in unit tests, and is thus required to build the BSP code base; but it is not intended for inclusion in the "standard" set of secured-BSP policies located in the directory indicated in (a) above.*
2. Policies that pertain to services OTHER THAN those described in (1) above are versioned at ${REPOSITORY_ROOT}/platform-config/trunk/policies/

Deploying Policies (rendering policies 'in effect')

To put policies into effect, they must be in a directory on the filesystem of the host running the BSP. The location of this directory is:

```
${BSPLOCALSTORE_HOME}/policies
```

Where BSPLOCALSTORE_HOME is assumed to be set in the context in which FUSE ESB (ServiceMix) is running. See the page *Developer Workbench Environment for BSP Service Developers* re: setting environment variables.

All policies must be located in the specified directory. No nested directories are read. The policies must be in place when the BSP's Policy service bundles are deployed to ServiceMix.