*I*nternational

*V*irtual

*O*bservatory

*A*lliance

Access Control Interface

Version 0.1

GWS Internal Draft 2008 May 07

This version:
ThisVersion-20080507
Latest version:
http://www.ivoa.net/Documents/latest/latest-version-name
Previous version(s):
None

Author(s):
Matthew Graham

Abstract

The Access Control Interface is an interface for managing access rights associated with a resource or set of resources. It specifies a set of methods for defining collections of users, resources, and operations and associating/dissociating them. It also specifies how access policies can be associated with users, resource, and operations.

Status of This Document

This is an Internal Draft within the Grid and Web Services Working Group. The first release of this document was 2008 May 07.

Conformance related definitions

The words “MUST”, “SHALL”, “SHOULD”, “MAY”, “RECOMMENDED”, and “OPTIONAL” (in upper or lower case) used in this document are to be interpreted as described in IETF standard, RFC 2119 [RFC 2119].

The **Virtual Observatory (VO)** is a general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The **International Virtual Observatory Alliance (IVOA)** is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications. The International Virtual Observatory (IVO) application is an application that takes advantage of IVOA standards and infrastructure to provide some VO service.

Contents

1 Introduction

Controlling who has access to resources and what operations are permitted on such resources is a common activity across the VO, e.g. editing resource records in the registry, deleting data objects in VOSpace, accessing proprietary data exposed through DAL services. Even though the actual resources and operations themselves will vary widely and different systems will be employed to enforce access rights, a common interface can be specified that sits on top of these different implementations, providing all the attendant benefits of interoperability, etc.

This document defines the Access Control Interface (ACI) for managing access rights associated with a resource or set of resources and the operations permitted on them in terms of identifying the methods that an instance of the interface must implement. This document does not specify any particular set of operations nor how any particular operation or set of operations would be represented. It also does not specify how access rights are represented within security credentials for the purposes of authorization.

1.1 Document roadmap

The rest of this document is structured as follows:

Section 2 describes the requirements for the ACI in terms of general, service output, flow and behaviour and interface details.

Section 3 specifies the methods associated with the ACI.

Section 4 illustrates how the ACI would be used in practice.

1.2 Terms in the document

The ***ACI*** is an abstract interface definition that specifies a set of methods.

An ***ACI instance*** is a deployed implementation of the ACI.

A ***service*** is any piece of VO software that has resources and is accessible via the Web. This can refer to a registry service, a DAL service (cone search, SIAP, SSAP, etc.) or a generic web service, both SOAP-based and RESTful.

An ACI instance is ***associated*** with one or more services when it is managing access rights to the resources that these services expose.

2 Requirements

The specification of the ACI has the following requirements.

2.1 General

- An ACI instance should be runnable as an interactive service that can be associated with any or multiple services. It should not need to reside on the same server as the service(s) with which it is associated.

- The interface should not reflect any dependency on any specific service implementation technology, e.g. SOAP. Different ACI instances can be defined for different implementation technologies.

- The interface should be compatible with IVOA authentication methods.

- The interface should not make any requirements for an ACI instance to have to maintain state about any transaction it performs.

2.2 Service Output

- The interface will return the permissibility of a set of operations for an identified resource or set of resources when requested.

- The interface will return the set of operations that an identified user or set of users is allowed to perform when requested.

- The interface will return the end goal status of any action to change the permissibility of a set of operations by an identified user or set of users on an identified resource or set of resources.

2.3 Flow and Behaviour

TBW

2.4 Interface Details

- The interface will consist of a set of methods that allow the access rights of resource records to be altered.

- The interface may allow a collection of resources to be defined. If the associated service does not support this activity, the ACI instance may maintain such state as is required to support this activity during a transaction.

In these cases, the ACI instance will not have to persist such state after the transaction is complete.

- The interface may allow a set of users to be defined. If the associated service does not support this activity, the ACI instance may maintain such state as is required to support this activity during a transaction. In these cases, the ACi instance will not have to persist such state after the transaction is complete.

- The interface should not disallow the use of a GUI with an ACI instance to facilitate interaction with the management methods.

3 Interface definition

3.1 Concepts

The ACI defines a data model that consists of the following concepts:

- An ***entity*** can be a user, a resource or an operation.

- A ***token*** is a tag that is used to identify an entity. This can take the form of a URI.

- A ***resource*** is identified by a globally unique identifier in the form of a URI. This need not be resolvable.

- A ***user*** is identified by a token. This may take the form of the DN component of a X.509 certificate or a URI but must be resolvable.

- An ***operation*** is identified by a token. This can take the form of a URI but must be resolvable.

- There can exist an ***association*** between a resource, an operation and a user. This means that the user can perform the operation on the resource. This may be identified in some fashion.

- A ***set*** or ***collection*** of resources, users or operations is identified by a token. This can take the form of a URI but must be resolvable.

- A human-readable ***policy*** can be associated with an operation. This provides a description of what the operation capabilities and constraints are and how they might be pertain to different classes of user. This will be identified with a URI.

3.2 Methods

The ACI specifies the following methods:

3.2.1 defineCollection

Define a set of resources, user or operations.

3.2.1.1 Parameters

- *collection:* the token for the collection
- *entities:* a set of tokens for the component entities

This associates the specified collection token with the specified set of entitites: resources, users or operations.

If no collection token is specified, the interface will return an autogenerated one.

The specified set must be homogeneous, i.e. it must only contain entities of the same type: resources, users or operations.

If the method completes successfully and a token was specified then it will be returned.

Once this method has completed successfully, the token can be used as a reference to the specified set.

3.2.2 getCollection

Get a list of tokens of entities in a collection.

3.2.2.1 Parameters

- *collection:* the token for the collection

This will return the tokens of the entities that comprise the collection identified by the specified token.

3.2.3 getOperations

Get a list of available operations.

3.2.3.1 Parameters

- *token**:*** the token for a collection or an entity
- *resource:* the identifier for a specific resource or set of resources

If no parameters are specified, this returns the full list of operations that the interface supports.

If the token parameter refers to a resource then the list of operations supported by the identified resource(s) is returned. If the token parameter refers to a set of resources then the common set of operations will be returned, i.e. all operations that can be performed on all the specified resources. If the token parameter refers to a user then the operations permitted for that user are returned. Finally if the token parameter refers to a set of users then the common set of operations permitted for those users will be returned, i.e. all operations that they are allowed to perform on all resources.

If both parameters are present, this returns the list of operations that are permitted for the specified user(s) on the specified resource(s).

3.2.4 getOperationDescription

Get a description of the operation.

3.2.4.1 Parameters

- *operation*: the token for an operation

This returns a human-readable description of the operation identified by the specified token.

3.2.5 associate

Associate a user, a resource and an operation.

3.2.5.1 Parameters

- *user*: the token for a user or set of users
- *resource*: the token for a resource or set of resources
- *operation:* the token for an operation or set of operations

This associates a user(s) with a resource(s) and an operation(s). It may return a token identifying the association.

3.2.6 dissociate

Dissociate a user, a resource and an operation.

3.2.6.1 Parameters

- *user:* the token for a user or set of users
- *resource:* the token for a resource or set of resources
- *operation:* the token for an operation or set of operations

This dissociates a user(s) from a resource(s) and an operation(s).

3.2.7 setPolicy(URI, token)

Associate a policy with an operation.

3.2.7.1 Parameters

- *policy*: the identifier for the policy
- *operation*: the token for an operation or set of operations.

This associates the identified policy with specified operation or set of operations.

3.2.8 getPolicy

Get a list of policies associated with an operation.

3.2.8.1 Parameters

- *operation*: the token for an operation or set of operations

This returns the set of identifiers of policies associated with the specified operation or set of operations.

4 Examples

These illustrate how the ACI could be used.

4.1 Supported operations

We want to see what operations are supported:

Request: getOperations()
Response: create, retrieve, update, delete

‘create’, ‘retrieve’, ‘update’, and ‘delete’ are just tokens identifying operations and should not be taken to mean anything more. If we actually wanted to understand what one of these actually meant then we would do:

Request: getOperationDescription(create)
Response: This operation will create a new resource

4.2 Defining collections

Next we want to group ‘create’, ‘retrieve’, and ‘update’ together as a set of operations identified as ‘cruPerm’:

Request: defineCollection(cruPerm, [create, retrieve, update])
Response: cruPerm

We also want to group all operations together as a set:

Request: defineCollection(crudPerm, [create, retrieve, update, delete])
Response: crudPerm

Similarly we can create a set of users:

Request: defineCollection(myGroup, [user1, user2, user3])
Response: myGroup

and a collection of resources:

Request: defineCollection(someResource, ivo://resource2)

Response: someResource

4.3 *Associating*

Finally we can give user1 full permissions on a resource and the rest of myGroup the ability to ‘create’, ‘retrieve’ and ‘update’ a resource (whatever that might mean - these are just tokens for the operations) but not to ‘delete’ one.

Request: associate(myGroup, someResources, cruPerm)
Response: success

Request: associate(user1, someResources, crudPerm)
Response: success

References