International
Virtual
Observatory
Alliance

Group Membership Service

Version 1.0

IVOA Working Draft 20181025

Working group
Grid and Web Services

This version
http://www.ivoa.net/documents/GMS/20181025

Latest version
http://www.ivoa.net/documents/GMS

Previous versions
This is the first public release

Author(s)
Brian Major, Patrick Dowler, Giuliano Taffoni, Adrian Damian, Marco Molinaro

Editor(s)
Brian Major

Abstract

The Group Membership Service (GMS) specification describes a REST interface for determining whether a user is a member of a group. This information can be used to protect access to proprietary resources: clients can issue a call to GMS when an authorization decision needs to be made. Proprietary resources can be any number of things such as data, metadata or services. Because a single group can be used to protect multiple resources, GMS enables the creation of groups that represent teams with common authorization rights. GMS offers organizations an interoperable, flexible and scalable way of protecting a hetergenous set of resources.

Status of This Document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Contents

Acknowledgments

???? Or remove the section header ????

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 RFC2119 (Bradner, 1997).

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.

1 Introduction

Through standard IVOA protocols, many astronomy data centres and institutes offer users access to datasets (DALI (Dowler, Demleitner, Taylor and Tody, 2013), Datalink (Dowler, Bonnarel, Michel, Donaldson and Languignon, 2013), etc), metadata (TAP (Dowler et al., 2010)) and storage (VOSpace (Graham et al., 2018)). In some cases this information is proprietary–it is only allowed to be accessed by certain individuals. Due to the wide variety and inherently institute-specific set of rules that may define how the information is proprietary, it is beneficial to the owners and maintainers of the rules to have a standard way of describing who has access to what resources. Additionally, the rules describing resource access may be determined by an entity external to the holder of these resources. To these ends, this document sets out a standard, programatic, and interoperable method of determining whether a given user is allowed to access a given resource.

The ideas presented by GMS enable data centres to do authorization checks in an interoperable fashion. In the context of authorization, interoperability can viewed on two levels: interoperability amongst the cooperating services *within* a data centre, and interoperability *between* data centres. Because of the orthogonal nature of authorization, these levels amount to the same problem.

Interoperability aside, GMS describes a simple, general, maintainable, and scalable approach to performing authorization, and so is a recommended architectural pattern for managing access to proprietary resources.

1.1 Proprietary resources

Most facilities have a period of time in which only the Principal Investigator's team has access to observation metadata and data files. Even without a proprietary period, time is required to verify and validate observations before they can be made public.

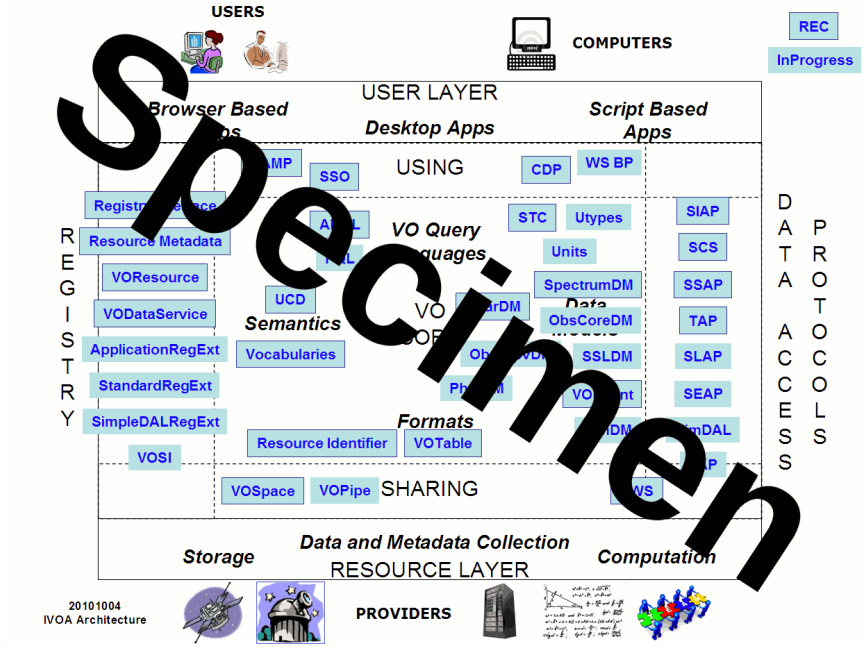

Figure 1: Architecture diagram for this document

Projects also frequently create higher level products such as catalogs and images. When these products are stored in a data centre, they must be accessible to only those who are authorized.

Proprietary information exists. For it to be made available in a data centre to those with authorization, a way of performing that authorization check is required.

1.2 Role within the VO Architecture

Fig. 1 shows the role this document plays within the IVOA architecture (Arviset et al., 2010).

GMS can be used by any software that needs to check, for authorization purposes, whether a user is a member of group. Because of this general use case, GMS cuts through all of the IVOA and lies squarely in the middle of the SHARING technical resource in the IVOA architecture diagram.

1.3 Use Cases

Asside from the main use case of restricting access to proprietary resources, GMS supports a number of other use cases, of both the user and system variety.

Proprietary information Restricting access to proprietary resources to certain users.

Homogeneity Using the same mechanism to control access to proprietary resources in a data centre or in multiple data centres.

Scalability A distributed mechanism that scales linearly with the resources being protected.

Remotely managing access A project may wish to control access to resources that reside externally.

Access rule sharing A project may consist of a variety of resources that can be all managed by the same access rules.

Extending the services of a data centre A project that has hosted data and metadata at a data centre may wish to create value-added services outside of the data centre itself. If some of the data or metadata is proprietary, the extended services may need to determine if a user is allowed to perform certain action on that data or metadata.

Cooperating institutes Two or more institutes may work together on a single project that involves proprietary resources so require a common mechanism for protecting those resources.

1.4 Definitions

Authentication User identification through credentials or identity provider. See IVOA Single-Sign-On Profile: Authentication Mechanisms. (Grid and Web Services Working Group, 2008)

Authorization Making the decision of whether to grant a user permission to a given resource. The decision can involve knowing the user's identity.

Resource Something that may require authorization for access. For example, a service, a data file, metadata.

User An individual identified by authentication.

Group A set of users.

Grant Authorizing access to a protected resource by assigning a group.

Revoke Removing access to a protected resource by removing an assigned group.

Owner A user or group of users who may grant or revoke access to a specific resource.

2 Authorization Requirements

When looking at a system that has proprietary resources that need to be protected, it is clear that there are two distinct phases to authorization: the assignment of the rules protecting the resources, and the attempts by various users to gain access to those resources. They are described here:

1. The owner(s) of a resource may, at any time, change the rules by which a resource may be accessed. This is the *granting and revoking of access*.
2. When users try to access resources, the granting rules for that resource are evaluated at runtime. This is the *authorization check*.

With these phases in mind and with the use cases defined, we can state that the goals of authorization are to:

- To allow for restricted access certain resources: only a certain set of individuals may access certain resources.
- To allow certain individuals to set the access rules on resources. The owner(s) of the resources need to manage the access rules.
- To be able to re-use granting rules between resources. Projects must authorize access to a variety of proprietary resources.
- To be able manage granting rules at a single location. Projects should not have to update each resource on a change to a re-used grant.
- To be able to reference remote granting rules. Proprietary resources should not be confined to a single institution.

3 Groups

3.1 Why Groups?

Why are groups a good model for authorization? When a system needs to perform an *authorization check* on a resource, it is trying to determine if the authenticated user is allowed access. There are a number of options on how this can be accomplished.

A simple approach would be to add the identity of the user to the resource. However, this is too restrictive as there may be multiple users who are allowed access. So, we could instead add a list of user identities to the resource being protected. It becomes a problem when there are two resources that need protecting by the same set of individuals. This becomes difficult to maintain because a change in access rules (*granting and revoking access*) would mean a change to multiple resources.

So, it becomes clear that this list of users needs to be decoupled from the resource so that it can be referenced and shared by multiple resources. To do so, the list must become a single entity than can be referenced by a name. And so, we must now have a named group of users.

A central repository of groups of users would introduce other problems: a single point of failure, and the inability to partition groups of users. Thus, the *location* of the group must accompany the group reference so that it is possible to have multiple collections of groups of users and multiple assiciated GMS services.

Resources must then reference a group by a URI with a location and a name that is unique within that location. This is called the Group Identifier.

Systems must use the information in the group identifier to query location to determine if the user is a member of the group. Because the location may be outside of the immediate vicinity of the resource, this query must be performed in a standard and accessible manner and so is defined as a RESTful interface to group membership.

3.2 Group Identifiers

A Group Identifier is an IVOID ((Demleitner et al., 2015)) used to uniquely and universally identify groups. They are attached to proprietary resources as grants. When it is time to determine group membership, the Group Identifier is resolved to a particular instance of a GMS service and a check for membership call is made.

For GMS, there are four important components to the Group Identifier:

1. The *scheme*: Always *ivo* indicating it is an interoperable IVOA ID.
2. The *authority*: This identifies the location or instance of the group membership service.
3. The *path*: Always starting with '/gms', indicating that it is a group URI, with '/instance-name' which is the name of the particular GMS instance within the authority.
4. The *query*: Identifies the group within the authority and instance. The name of the group.

Below is an example group identifier:

```
ivo://authority.example.com/gms/instance1?groupName
```

To resolve the host GMS service URL, one would issue a query to RegTAP ((Demleitner et al., 2013)) to find the accessURL in the interface for the authority. The following query will return a row for each *access_url* and *security_method_id* combination. The ivoid value is calculated by removing the query string from the group identifier. Since we are looking to perform a is member call, we ask for the GMS search capability, identified by the GMS search standardID (see section below).

```
SELECT access_url, security_method_id
FROM rr.interface
NATURAL JOIN rr.capability
NATURAL JOIN rr.resource
WHERE
  ivoid = 'ivo://authority.example.com/gms/instance1' AND
  standard_id = 'ivo://ivoa.net/std/gms#search-1.0'
```

This would result in an access URL capable of supporting a GMS search on the group 'groupName'. For example:

```
http://server.example.com/myGMSImpl/search
```

4 GMS Search API

4.1 API Definition

The Group Membership Service defines a RESTful API (Fielding, 2000) that allows for the determination of whether a user is a member of a group. This is the GMS search capability and is identified by the following standardID:

```
ivo://ivoa.net/std/gms#search-1.0
```

Within this capability, there are two functions:

- *boolean isMember(Group, User)*: Return true if User is a member of Group.
- *list<Group> getMemberships(User)*: Return the list of Groups of which User is a member.

The resulting REST API for these functions is as follows:

```
GET /search/{group}
GET /search
```

Where *search* represents the *access_url* from the RegTAP call and *{group}* is the groupName part of a Group Identifier.

Two (optional) parameters can be supplied to identify the user:

```
user=<user-principal>
principal=<principle-type>
```

An HTTP GET to */search/{group}* shall respond with HTTP 200 (OK) if the user is a member of the *{group}*. If the user is not a member of the group, or if the user is not recognized, or if the group is not recognized, the service shall respond with HTTP 403 (Forbidden).

An HTTP GET to */search* shall return HTTP 200 (OK) with a list of the groupNames in which the user is a member in the response body. The response must have a Content-Type of *text/plain* and each group must be separated by a new line character. If the user is not a member of any groups, or if the user is not recognized, the response body must be empty.

The *user* and *pricipal* parameters are used to identify the user who is the subject of the membership question. The *user* field is the username of the user in context of the *principal* value. For example, when the *principal* field is set to 'X.509', the *user* field will contain the user's distinguished name. Or, if the *principal* field is set to 'OpenID', the *user* field would contain the user's OpenID token. For the full list of supported principal types please refer to the User Identification standard (**Note for authors**: This is to be written). If the GMS service does not recognize the value of the *principal* parameter, the service shall respond with HTTP 501 (Not Implemented). If the *pricipal* parameter is recognized but the user cannot be identified, the service shall respond with a HTTP 403 (Forbidden) (in a call to */search*) or with an empty list of groups (in a call to */search/{group}*).

If the *user* and *principal* parameters are not supplied, it is assumed that the user who is the subject of the membership question is the user who is making the REST call. This pattern will be in use when the call is being made by a service that supports and implements the IVOA Credential Delegation Protocol (Graham et al., 2010). If the user cannot be identified from the call because they have not authenticated, the service must respond with HTTP 400 (Bad Request). The other HTTP responses shall be the same as described above where the user was identified by the *user* and *principal* parameters.

If one of *user* or *principal* are supplied, then they both must be supplied. If only one is supplied then the service must respond with HTTP 400 (Bad Request).

(**Note for authors**: It could be that the *user* and *principal* parameters are turned into one parameter that is in URI format and contains enough information to identify the user across different authentication mechanisms.)

4.2 Search Examples

Example 1 - Group access to a VOSpace Node A user is trying to download a VOSpace file that has the group-read property set to

```
ivo://authority.example.com/gms/instance1?my-collaboration
```

This resolves (though a RegTAP query for the search API) to host

```
http://server.example.com/groupService/search
```

To authorize the user, the VOSpace service queryies the GMS search service using the user's delegated credentials

```
HTTP GET to http://server.example.com/gmsService/search/my-collaboration
```

The GMS service identifies the user, consults it's groups memberhip information, and returns a response code of 200 when confirming the user is a member of group 'my-collaboration'.

Example 2 - Group access to table data A user issues an ADQL query to table with row-level authorzation in a TAP service. A read-group column defines which group is allowed to read that row. The first row that is encountered with a non-null read-group has value:

```
ivo://authority.example.com/gms/instance1?my-other-collaboration
```

Not wanting to make a REST call for each row that needs to be consulted, the TAP service asks for all groups for the user on the first group protected row that is encountered. The service does not have the user's delegated credentials, so passes the user information in the search call.

```
HTTP GET to http://server.example.com/gmsService/search?user=bmajor&pricipal=username
```

The GMS service returns HTTP 200 and all the groups in which user 'bmajor' is a member:

```
my-collaboration
my-other-collaboration
my-final-collaboration
```

The TAP services caches this group membership information for the lifetime of the request so that it can be used if necessary when checking other rows. If a read-group entry with a different IVOID is encountered, the TAP service must call that GMS service too and add the list of groups to its cache.

5 Implementation

5.1 Implementation Options

- Via Grouper (groups in MySQL, users in LDAP)
- LDAP only with memberOf plugin (supports groups-of-groups)
- VOSpace implementation: ContainerNodes = groups, DataNodes = users

5.2 User Identity

The concept of users and user identity is core to group authorization. When a system makes a call to a GMS service to determine if the user trying to access the resource is a member of a group, the GMS service needs to identify that user with the users in various groups.

(Author note: add reference or table of user identity types.)

The collection of data centres and astronomy institutes likely have many ways of identifying users. They could be using external identity providers, they could have a local database of users, or may have a combination of these and other approaches. This specification does not require such a design. Instead, it requires simply that users can be uniquely identified within the scope of a GMS service's domain. If a user identity reaches beyond the scope of a GMS service's domain (such as an X.500 distinguished name (**?**)), then it, too, may be referenced by the service.

5.3 Information Privacy

User and group membership information may be private, so who is allowed to make GMS search calls must be considered when implementing a GMS service. A GMS implementation may insist that GMS search calls must be made by a certain privileged account only. This is a reasonable approach when the service is only used with a single organization, but would require the distribution of those privleged credentials to any external sites wishing to use it.

Alternatively, a GMS service could have a policy where only the user who is the subject of the membership assertions is allowed to make the GMS search calls. This approach lends itself well to external interoperability because there need not be any sharing of credentials or trust arrangements between sites–it is always only the user who makes the service calls, even when they are transitive. This is the approach recommended in the IVOA credential delegation protocol (Graham et al., 2010). So, aside from the architectural benefits of employing this pattern, there are some information privacy concerns that are addressed.

5.4 Groups of Groups

It may be functionally attractive to support groups within groups. If this is implemented, then the service must ensure that this representation is reflected by the service API. For example, if an isMember(g) call is made, and the group 'g' is a group within another group in which the user is a member, then the service must return true. The fact that the service supports groups within groups is not exposed through the search API, but the API does not prohibit such an implementation.

If one of the contained groups actually exists at another GMS instance, perhaps outside of the organization, then the service must transitively query that service to determine group membership.

A Changes from Previous Versions

No previous versions yet.

References

Arviset, C., Gaudet, S. and the IVOA Technical Coordination Group (2010), 'IVOA architecture', IVOA Note.
URL: *http://www.ivoa.net/documents/Notes/IVOAArchitecture*

Bradner, S. (1997), 'Key words for use in RFCs to indicate requirement levels', RFC 2119.
URL: *http://www.ietf.org/rfc/rfc2119.txt*

Demleitner, M., Harrison, P., Molinaro, M., Greene, G., Dower, T. and Perdikeas, M. (2013), 'IVOA registry relational schema', IVOA Working Draft.
URL: *http://www.ivoa.net/documents/RegTAP/*

Demleitner, M., Plante, R., Linde, T., Williams, R. and Noddle, K. (2015), 'IVOA identifiers, version 2', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/REC/Identifiers*

Dowler, P., Bonnarel, F., Michel, L., Donaldson, T. and Languignon, D. (2013), 'Datalink', IVOA Working Draft.
URL: *http://www.ivoa.net/documents/DataLink/*

Dowler, P., Demleitner, M., Taylor, M. and Tody, D. (2013), 'Data access layer interface, version 1.0', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/DALI*

Dowler, P., Rixon, G. and Tody, D. (2010), 'Table access protocol version 1.0', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/TAP*

Fielding, R. T. (2000), Architectural Styles and the Design of Network-based Software Architectures, PhD thesis, University of California, Irvine.
URL: *http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm*

Graham, M., Morris, D., Rixon, G., Dowler, P., Schaaff, A., Tody, D. and Major, B. (2018), 'VOSpace specification - version 2.1', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/VOSpace/*

Graham, M., Plante, R., Rixon, G. and Taffoni, G. (2010), 'IVOA credential delegation protocol'.
URL: *http://ivoa.net/Documents/CredentialDelegation/20100218/*

Grid and Web Services Working Group (2008), 'IVOA single-sign-on profile: Authentication mechanisms version 1.01'.
URL: *http://www.ivoa.net/documents/latest/SSOAuthMech.html*