Group Management Use Cases

August 1, 2019

Version 1.2

These use cases describe how individuals need to manage and use group definitions. In its simplest form, a group definition is *a list of people* that needs to be maintained and used for some ongoing purpose. GRP-02 through GRP-16 were originally derived from a project-wide user needs study conducted in early 2017 by the XSEDE Community Infrastructure (XCI) team's Requirements Analysis and Capability Delivery (RACD) activity. [1] However, these use cases are consistent with group features in many systems, so their usefulness is by no means specific to XSEDE.

GRP-01: Researcher manages membership of a project group

A **researcher** wants to change the membership of a project group. We assume the researcher has registered with the community and is a manager of a project that has been granted an XSEDE allocation.

In most cases, the researcher wants to experience it like the following steps.

1. First, the researcher logs into the community portal.
2. Then, the researcher clicks Add user and selects the project to be managed.
3. Then, the researcher uses the user interface to add or remove individuals to the project membership.

We'll take any solution, as long as the following are true.

1. If the researcher has multi-factor authentication enabled, Step 2 will require the researcher to authenticate via a second factor before proceeding.
2. The system must support O(25) researchers going through this workflow simultaneously, under the assumption that individual researchers may walk away from their browser during the process and leave it hanging.
3. The "Add user" interface must be available and working as described at least 99.9% of the time.

GRP-02: Manually create a group

A **community member** needs to create a group via a simple user interface. We assume that the community member has already registered with the community.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member logs into the community portal.
2. Then, the community member navigates to the "Groups" section.
3. Then, the community member clicks a button to create a new group.
4. Finally, the community member fills out a brief form setting the initial group configuration, including a name for the new group and basic access control settings, including the option of requiring multi-factor authentication for future changes. The community member is automatically assigned the administrator role within the group.

It will always be like this except when, in Step 4, the community member sets the initial configuration to require multi-factor authentication for group management. In that case, the interface will require multi-factor authentication to create the group.

We'll take any solution, as long as the following are true.

1. If the community member has multi-factor authentication enabled, Step 3 requires multi-factor authentication before proceeding to Step 4.

2. In Step 4, the community member should be able to give the new group any name, though there may be restrictions on how the name is constructed. (Allowed/prohibited characters, etc.)
3. In Step 4, the basic access control settings should include: (a) is the group searchable by non-members; (b) who can see group members; (c) who can see role assignments for group members; (d) are members required to supply any additional profile information?
4. In Step 4, a flag is set in the group's definition indicating that multi-factor authentication for group management was/was not required since the group's creation.

GRP-03: Manually view or manage the configuration or membership of a group

A **community member** needs to view or change the configuration or membership of a group via a simple user interface.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member logs into the community portal.
2. Then, the community member navigates to the "Groups" section.
3. Then, the community member selects the group via a search or browse function.
4. The group's configuration and membership is displayed.
5. Finally, if the group's access control settings and the community member's role in the group allow it, the community member can use the user interface to change the group's configuration and add or remove individuals in the group membership.

It will always work like this except in the following cases.

1. If, in Step 3, the group's access controls restrict visibility of the group and the community member does not satisfy the required criteria, the group will not appear in search results or browsing lists.
2. If, in Step 4, the group's access controls restrict membership visibility and the community member does not satisfy the required criteria, the interface will not display the group's membership.
3. If, in Step 5, the community member does not have the administrator or manager role in the group, the interface will not present the option to change the group's configuration or membership.
4. If, in Step 5, the community member disables multi-factor authentication for group management, the interface should warn the user before saving changes that this will make the group ineligible for use in applications that require multi-factor authentication since group creation.

We'll take any solution, as long as the following are true.

1. The configuration settings that can be viewed and changed in Step 5 include: group name, owner, members and roles, basic access control settings, and synchronization settings (see GRP-9).

2. Newly added group members are given the “member” role and no others, unless the interface allows the community member to explicitly assign a different role.
3. If the group configuration requires multi-factor authentication for group changes, Step 5 requires multi-factor authentication.
4. If the community member has multi-factor authentication enabled, Step 5 requires multi-factor authentication.
5. If the community member disables multi-factor authentication for group management, the flag indicating that multi-factor authentication has been enabled since group creation time (see GRP-02) is set to “false.”

GRP-05: Invite members to a group

A **community member** needs to invite one or more individuals (“invitees”) to become members of a group. The invitees will not become group members until they accept the invitation. The community member must have the administrator or manager role in the group.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member logs into the community portal.
2. Then, the community member navigates to the “Groups” section.
3. Then, the community member finds or selects the group via a search or browse function.
4. Then, the community member chooses the “invite” function.
5. Then, the community member uses the user interface to search for or manually enter the names or email addresses of the invitees and the system sends email messages to the invitees with a link to click to accept the invitation.

The **invitees** want to experience it as follows.

6. First, the invitee receives an email message with a link to click to accept the invitation. In order to proceed, the invitee must click the link.
7. If the invitee does not already have an active login session in the community portal, the invitee will be asked to login or register. In order to proceed, the invitee must login to an existing community identity or register and login.
8. Then, the invitee will be presented with a page displaying information about the group and the invitation and a confirmation button to accept the invitation. In order to proceed, the invitee must click the button.
9. Then, if the group’s basic access controls require members to supply additional profile information, the invitee will be asked to enter the information. In order to proceed, the invitee must enter the information.
10. Finally, the invitee will be added to the group and informed of the successful operation.

We’ll take any solution, as long the following are true.

1. If the group's configuration is set to require multi-factor authentication for group management, Step 4 will require multi-factor authentication.
2. If the community member has multi-factor authentication enabled, Step 4 requires multi-factor authentication.
3. In Step 5, the search interface respects privacy (particularly guarding against unintended release of email addresses) and protects against name collisions.

GRP-06: Request membership in a group

A **community member** ("requester") needs to request to be added to the membership in a group via a simple user interface.

In most cases, the **requester** wants to experience it as follows.

1. First, the requester logs into the community portal.
2. Then, the requester navigates to the "Groups" section.
3. Then, the requester selects the group via a search or browse function.
4. Then, the requester uses the user interface to request to be added to the group membership.
5. If the group's basic access controls require members to supply additional profile information, the requester will be asked to enter the information. In order to proceed, the requester must enter the information.
6. Finally, the requester will be informed that his/her request has been sent to the group managers for processing.

The **group manager** wants to experience it as follows.

7. First, the group manager will receive an email message with information about the request and a link to click to approve/reject the request. In order to proceed, the manager must click the link.
8. If the manager does not already have an active login session in the community portal, the manager will be asked to login or register. In order to proceed, the manager must login to an existing community identity or register and login.
9. Then, the manager will be presented with a page displaying information about the requester and an interface to accept or reject the invitation. In order to proceed, the manager must accept or request the invitation.
10. Finally, if the manager accepts the request, the requester will be added to the group, the manager will be informed of the successful operation, and the requester will be informed of acceptance via email.

We'll take any solution, as long as the following are true.

1. In Step 3, the group should only be visible to the community member if the group's basic access control settings allow it based on the requester's role(s) in the group.

2. In Step 4, the interface to request membership in the group will only be available to the requester if the group's basic access controls allow membership requests.
3. If the community member has multi-factor authentication enabled, Step 4 requires multi-factor authentication.
4. Newly added group members are given the "member" role and no others, unless the interface allows the manager to explicitly assign a different role.
5. If the group's configuration is set to require multi-factor authentication for group management, Step 9 will require multi-factor authentication.

GRP-09: Synchronize an external group

A **community member** needs to synchronize a group's membership with a group defined in another system so applications that use groups for access control (use cases GRP-12, -13, -14) can reference a group that is primarily managed in another system. E.g., a multi-institution research team uses the Open Science Grid's VOMS service to define its membership, and an XSEDE L3 service provider at Indiana University wants to authorize jobs submitted by the team. The IU resource is already integrated with XSEDE's group system, so it would be less expensive and less complicated to use an XSEDE group than to install a local VOMS instance and use it for access control. [1]

In the ideal case, the **community member** would experience it as follows.

1. First, the community member logs into the community portal.
2. Then, the community member navigates to the "Groups" section.
3. Then, the community member selects the "synchronize" function.
4. Then, the community member specifies an external group service, the name of the group in the external service, a name for the community's copy of the group, and synchronization options.
5. Then, the system attempts to automatically map the members of the external group to known community identities. It displays any members of the external group who cannot be automatically mapped to community identities and the community member is given the option to ignore the member, to send an email invitation to the group to the member, or to supply a community identity for the member (via a search).
6. Finally, the system notifies the community member that the group has been successfully created and synchronized with the external group.

We'll take any solution, as long as the following are true.

1. If the community member has multi-factor authentication enabled, Step 3 requires multi-factor authentication.
2. In Step 4, the external services supported ideally include: Google, OSG/VOMS, Grouper, COmanage, Globus, HUBzero.
3. In Step 4, if the community member sets the initial configuration to require multi-factor authentication for group management, the interface will require multi-factor authentication to proceed.

4. The group's flag indicating that multi-factor authentication has been required for group management since creation time is set "false," even if that requirement is specified by the community member in Step 4. (Externally synchronized groups should not be eligible for use in applications that require multi-factor authentication.)
5. In Step 5, the community member is automatically given the administrator role for the new group.
6. In Step 5, the system keeps track of email invitations and/or local identity mappings so that subsequent synchronization actions maintain the current mappings.
7. If the new group is subsequently looked up in the system (e.g., use case GRP-03), it is clear that the group is synchronized with an external group, and the synchronization settings (including member identity mappings) are visible.
8. Whenever the group is synchronized with the external group service, if there are any new group members who cannot be automatically mapped to a known identity, the system sends an email report to the group administrator(s) informing them of the issue and they are given a way to specify identity mappings as described in Step 5.

GRP-10: Automate a group's configuration

An **application developer** or **system administrator** (hereafter referred to as the "group user") needs to enable an application (or science gateway) to access and/or manage community groups. In other words, we need APIs for use cases GRP-02, -03, -05, and -06 in addition to a human interface.

In most cases, the **group user** wants to experience it as follows.

1. First, the group user visits the community website (or a website for application developers) and searches for documentation on group management interfaces.
2. Then, the group user registers his/her application with the community and obtains credentials that allow the application to use the group management API(s).
3. Then, the group user selects an existing software development kit (SDK) or a locally installed client tool (CLI) matched to the development environment he/she is using from a list in the documentation, configures it with the credentials obtained in Step 2, and uses it to develop code in his/her application.
4. Finally, when the application is deployed and used by researchers, groups defined and managed in the application are actually created and managed in the community group system and are accessible by the application, other applications, and other services as described in the GRP-* use cases.

It will always be like this except when the group user chooses to use a CLI, in which case Step 2 is not required, the credential configuration described in Step 3 is not required, and in Step 4, the group user must authenticate at least once to use the CLI. (The CLI must be able to retain and reuse credentials after a successful authentication.)

We'll take any solution, as long as the following are true.

1. The documentation in Step 1 is openly available without requiring the developer to identify or authenticate him/herself.
2. The registration in Step 2 uses the standard community login mechanism.
3. The development environments for which SDKs are available in Step 3 ideally include: Python, Java.
4. The execution environments for which CLIs are available in Step 3 ideally include: Linux, macOS, Windows.
5. The API is standardized beyond the current public research computing community.
6. The API, CLI, and at least some of the SDKs are freely available for academic and research use.
7. The API, CLI, and at least some of the SDKs are in widespread use.
8. The API, CLI, and at least some of the SDKs are supported by external sources.
9. The API, CLI, and at least some of the SDKs are available in open source form.

GRP-12: Use groups to control access within a resource

A **service provider** needs to use groups to control access to his/her resource. The most common reason is to limit access to the resource to project groups that have active allocations via an official allocation process. We assume that the groups to be used by the service provider have already been created and populated in the group management system.

In most cases, the service provider wants to experience it as follows.

1. First, the service provider visits the community documentation for service providers, reviews the documentation on how to use group data, and obtains the client tools necessary to access group data. (See GRP-10.)
2. Then, the service provider develops a local mechanism for integrating the client tools with the resource's account management and accounting mechanisms. This typically involves using the data supplied by the group service to configure local accounts, local groups, filesystems and other local resources, quotas, accounting/usage reporting systems, etc.
3. Finally, the service provider monitors logs and other status services to ensure that ongoing changes to group data result in the appropriate local system configuration changes and to resolve any failures or exceptional conditions.

We'll accept any solution, as long as the following are true.

1. In Step 1, the client tools can be used on most Linux and other Unix-like systems.
2. In Step 2, the client tools and the group configurations (specifically, user-defined attributes in the group configurations) are sufficient to enable service providers to identify and select groups to be used locally based on attributes. For example, an allocations administrator could create allocation groups using a specific administrator identity and with specific user-defined attributes that specify (a) that the group is an allocation group, and (b) that the allocation is for a specific SP resource or service. A service provider would then be able to select groups relevant to his/her resource based on those attributes and would furthermore be able to inspect the group's basic access controls to ensure that the group definition can be trusted. Service

providers must be able to access the flag indicating whether or not multi-factor authentication has been required for group management since the creation of the group. (See GRP-02, -03, and -09.)

3. In Step 2, the client tools enable service providers to "mirror" the data for their system(s), populating the local system with a complete copy of all relevant data available from the source.
4. In Step 2, the client tools enable service providers to receive changes to group data on a continuous basis.

GRP-14: Use a group to control access within an application

A **community member** needs to use a community group to control access within an application. Notable reasons include: a **campus or L3 service** offers services to specific community members, a **science gateway** offers group-based access control to gateway functions or resources, a project member limits access to a project tool (or administrative privilege within a project tool) to members of a project's WBS group. The application must either have a "plugin" feature that allows it to use an external group service, or a "local file" feature that allows it to use local files containing user IDs as the basis for access control.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member accesses the group management interface, either in the community portal (as described in use cases GRP-02, -03, or -09) or within the application itself (if it has been integrated as described in use case GRP-10).
2. Then, the community member creates and/or configures a group to be used within the application.
3. Finally, the community member configures the application's access control either by selecting the group in the application's user interface or by referring to a local text file containing the group's membership data that was created & maintained using the CLI described in GRP-10.
4. Any subsequent changes to the group in the group management service are automatically reflected in the application's settings without additional human intervention.

It will always be like this except when the group to be used in the application already exists, in which case Steps 1 and 2 are not needed.

We'll accept any solution, as long as the integration between the application and the group management service is accomplished as described in use case GRP-10.

GRP-15: Use a group for task assignments within an application

A **community member** needs to assign a task in an application to a group (as opposed to an individual). The application must either have a "plugin" feature that allows it to use an external group service, or a "local file" feature that allows it to use local files containing user IDs as the basis for access control.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member logs into the application.
2. Then, the community member selects a task via a browse or search function.
3. Finally, the community member uses the tool's interface to assign the task and selects a group using the tool's interface.

We'll accept any solution, as long as the following are true.

1. In Step 1, the login uses the standard community login mechanism.
2. In Step 3, the application's interface offers groups drawn from the community's group management service with current memberships.
3. In Step 3 or subsequent actions, if the application sends email to the group members, it uses the email address listed in their community user profile.
4. The integration with the community's group management service is accomplished as described in use case GRP-10.

GRP-16: Use a group for email distribution

A **community member** needs to use a group as the source for the members of an email distribution list. One notable reason for this is to avoid membership "drift" (changes over time) between a group used for access control and the corresponding email distribution list. E.g., a project maintains groups for project WBS areas. For each WBS area, they'd like a single group to be used both for access control in project applications and for email communications.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member creates and configures the group.
2. Then, the community member uses a tool developed as described in use case GRP-10 to set up a regular extraction of the email addresses for the group into a text file on the email distribution list server (most likely a mailman server or equivalent).
3. Finally, the community member configures the email distribution list software (again, most likely mailman or an equivalent) to use the file as the source of addresses for the group's email distribution list.

We'll accept any solution, as long as step 1 is accomplished via use cases GRP-02, -03, or -09.

GRP-17: Drive project membership with an email message

A **researcher** or **educator** wants to send the community's account management team a list of names and email addresses, resulting in everyone on the list receiving a community account and being added to the

researcher/educator's allocation project group.[1] The researcher/educator must be a manager for the allocation.

In most cases, the **researcher/educator** wants to experience it as follows.

1. First, the researcher/educator obtains an allocation. The notification of the allocation will include a project identifier.
2. Then, the researcher/educator sends an email message to the accounting team containing the project identifier and a list of names and corresponding email addresses for individuals who should be members of the allocation's project group.
3. Once the email message has been accepted by the accounting team, each person on the list who already has a community identity with the given email address is added to the project group. Email addresses that are not associated with community identities will be sent an invitation to log in with a community identity or register a new one. Once they've confirmed their community identity, it will be added to the project group.
4. Finally, the researcher/educator receives an email response with the initial status of each member of the list (either added to the project or sent a request to authenticate/register). He/she can check the membership of the project at any point to confirm that everyone has been added.

We'll take any solution, as long as the following are true.

1. Step 1 uses the normal community allocation mechanism.
2. In Step 2, the email must be sent from an email address associated with a PI for the allocation.
3. In Step 3, there should be some form of verification that the email message from the researcher/educator is authentic. This might, for example, involve sending a response asking the sender to login (requiring him/her to authenticate) and confirm the request. But it should not require the sender to re-submit information that was provided already.
4. In Step 3, if a listed email address is not associated with a community identity and the recipient of the invitation chooses to register a new identity (rather than log in with an existing community identity), there must be some form of verification that the newly registered identity is authentic before it is added to the project group. This might, for example, involve inspection by the accounting team or by the researcher/educator.

References

[1] User Needs Assessment: XSEDE-wide Group Features. Technical Report, April 6, 2017. (https://software.xsede.org/svn-public/xci/inputs/user-needs/XSEDE-wide-groups.pdf)

History

Version	Date	Changes	Author

[1] Original request: https://jira.xsede.org/browse/UR-218

Entire Document	1.0	5/2017	Drafted use cases 1-16 based on XSEDE project-wide needs assessment for group management functions [1]	Liming
Specific use cases	1.1	11/14/2017	Combined 1, 3, and 4 into new 3; deleted 7 and 8 by integrating them into new 3 and existing 9; integrated 11 with 10; added new 17	Liming
Entire document	1.21	08/01/2019	Removed unnecessary XSEDE terminology; merged GRP-13 into GRP-14.	Liming