Grouper Deployment Guide

Welcome to the Grouper Deployment Guide (GDG). Are you looking for a place to get you started with Grouper concepts, including the suggested standards that are being adopted across the community? If so, you've come to the right place.

GDG Table of Contents

What's new in the Grouper Deployment Guide?

Folder and Group Design
Jan 28, 2021 • updated by Chris Hyzer (upenn.edu) • view change

Access Control Models
Jun 10, 2020 • updated by Emily Eisbruch (internet2.edu) • view change

Folder and Group Design
Jun 10, 2020 • updated by Emily Eisbruch (internet2.edu) • view change

Get help

Can't find what you are looking for?

help Ask the community

Acknowledgements

This guide is the result of a community collaborative effort. The initial version was made possible by contributions from the TIER API and Entity Registry WG, the Grouper Development Team, and many others. This wiki supersedes Grouper Deployment Guide Version 1. The guide is also no longer managed under the Trust and Identity Document Stewardship process. The wiki version will continue to be updated as community practice and the Grouper product evolves.

Executive Summary

> *"It's not just about SAML federation, it's about enabling high-value collaboration across thousands of disciplines and millions of people. Hence agreement on attribute and authorization management, application integration, administration procedures, workflow, privacy management,..."*
> ***- RL "Bob" Morgan***

Access management capabilities in higher education and research tend to be a mix of institution specific custom solutions, whether they are built on in-house frameworks, proprietary closed-source "solutions", or open source toolkits like Grouper.

What if, instead of every institution having its own special sauce there was broad agreement on access management strategies, vocabulary, and assumed capabilities? What if, we could drive "federation" deeper into institutional Identity and Access Management (IAM) practices and more easily enable high-value collaboration across thousands of disciplines and millions of people?

Grouper provides distributed access control governance, fast flexible provisioning integrations, along with robust auditing and reporting to answer who, why, when, and how someone has access to a resource. The Grouper project maintains documentation and training materials on the Grouper wiki mostly in the form of administration guides, community contributions, and training videos. These materials do a very good job of providing reference materials, and a variety of deployment and use case examples. However, for the uninitiated it is not always clear where to start and how to stay on the right path. Additionally, many configuration choices and deployment options are left for the deployer to decide. This has led to deployments which have tended towards similar functionality, but often diverge considerably in approach, terminology and implementation.

This deployment guide distills a variety of community practices represented in group and folder design ideas and the various deployment examples in community contributions into a common community approved approach. Harmonizing Grouper deployments with common practice, vocabulary and IAM strategies will make it easier for the community to work together toward common objectives and improve Grouper more quickly over time. It will also enable new and existing Grouper deployments to more easily benefit from community experience, achieve access management goals more quickly, and work together to build robust IAM capabilities based on the InCommon Trusted Access Platform.

The goal of this guide is to help you come up to speed on Grouper concepts, how they relate to identity and access management, and how they can be deployed to implement effective access control in a wide variety of situations.

Previous: Grouper Deployment Guide

Next: Introduction to the GDG

Introduction to the GDG

Overview

An InCommon Trusted Access Platform (TAP) based identity and access management program deploys Grouper as a strategic component of its institutional role and access management solution. Grouper is at the center of all group-like management activities, such as institutional roles, access control lists, service eligibility, and email distribution lists.

Grouper can be employed in a variety of flexible access control models, but the underlying approach follows a consistent path:

1. Natural language access management policy drives requirements which help to identify and define institutional cohorts (types of students, types of employees, types of visitors/guests, etc.).
2. Institutional cohorts are then turned into reference groups which are used in the digital access policy definition.
3. Access to systems is then automatically kept in sync with policy as subject attributes change in underlying systems of record (ERP, SIS, etc).

This provides streamlined and automated access for existing and future applications.

In addition to automated access provisioning, Grouper supports user-defined populations. These groups can be self-administered, and attested to, by an authoritative delegate, or, allow members to join or leave as they choose. These ad-hoc populations can then be used in authorization policy as exceptions, self-service, or other forms of populations.

Purpose and Scope

> *"...some additional scaffolding in the form of configurations and conventions based on successful models at other campuses would accelerate an adopting campus's path to rolling out actual services."*
> *- Warren Curry, University of Florida*

This deployment guide aims to make it easier for new deployers to understand Grouper, complete an initial deployment, and implement access management capabilities based on common practice and terminology.

The guide focuses on access management governance using widely deployed Grouper primitives and features. Grouper can also be used to externalize fine-grained application-level permission management. However, that feature has not yet see significant adoption and is mostly out of scope.

Audience

This guide is primarily written for IT architects, technical staff and managers responsible for identity and access management. However, the guide would benefit anyone who is seeking information on an InCommon Trusted Access Platform compatible Grouper deployment.

Readers should be familiar with identity and access management concepts and terminology as defined by NIST Special Publication (SP) 800-162, *Gui de to Attribute Based Access Control (ABAC) Definition and Considerations,* the Grouper glossary, and Grouper UI terminology. Readers completely new to Grouper would also benefit from reviewing the video Intro to Grouper Pt. 1/3: Access Management & Grouper and attending the InCommon Grouper School.

Using the Grouper Deployment Guide

This rest of the deployment guide is organized as follows:

- Understanding Grouper - provides a basic understanding of Grouper and how it relates to the InCommon Trusted Access Platform.
- Folder and Group Design - defines a variety of group types and purposes, and a recommended initial folder and group organization.
- Access Control Models - describes how Trusted Access Platform and Grouper components come together to achieve access management capabilities.
- Provisioning Models - discussion on provisioning models and strategies.
- Operational Considerations - provides pointers and tips on operating Grouper in production.
- Conclusion - concludes the document
- Example Access Policies - provides example access policies.

The guide focuses on high level explanation and discussion. For current technical details and version specific information please refer to the Grouper wiki.

Terminology

This guide builds on and makes use of the terminology defined in NIST Special Publication (SP) 800-162, *Guide to Attribute Based Access Control (ABAC) Definition and Considerations,* for IAM level concepts, and borrows Grouper product specific terminology from the Grouper glossary and Group er UI terminology. When first exposed to Grouper, there is a tendency to view everything as a "group". This document adopts the following terminology to distinguish Grouper primitives (e.g. "groups") from IAM level concepts.

Grouper primitives are specific names for features within the product itself. These may be mapped to one or more IAM level concepts that are used to implement various capabilities. A **Grouper group** is a primitive and that can be used in many different ways to implement the desired access control mechanism.

Subjects that have been added directly to a group are said to have a **direct membership assignment**. Subjects that are members of a group by virtue of membership in another group are said to have an **indirect membership assignment**. That is, they are members because of their membership in a subgroup that is itself a member of the parent group. A **composite group** is the result of combining two other groups, typically by relative complement (i.e. Group A minus Group B).

In order to distinguish the intended use of a group this document will qualify the word “group”. For instance, a **reference group** is a named group of subjects that is largely intended to be used by reference within access policy groups. Reference group names can also be thought of as labels or tags that are applied to all the members of the group. In this way, they can also be viewed as subject attributes from an ABAC policy perspective.

Basis groups tend to consist solely of direct subject membership assignments and are often maintained automatically by the Grouper Loader process based on data from a system of record. Basis groups are typically subsets of cohorts that when used together in different combinations form proper reference groups. For instance, an HR system might have different codes for various employees. These cohorts might be loaded separately into basis groups and then combined into an “employee” reference group.

Access policy groups are the digital representation of the natural language requirements for specific access, permissions, or role in a target service. Access policy groups are often implemented as a composite group whose membership is composed of an allow group and a deny group. Effective membership in an access policy group represents a precomputed access policy decision. Membership within an access policy group may be kept in sync directly with a target service or an intermediary like an LDAP based enterprise directory service, and is often incorporated in a SAML authentication response via Shibboleth.

Controlling who has access to objects and actions within Grouper itself is control by Grouper Privileges. Grouper Privileges can be assigned to individual Grouper users. However, it is best practice to assign them to specific groups within Grouper. Groups used to assign Grouper Privileges to users are called **security groups.**

Previous: Executive Summary

Next: Understanding Grouper

Understanding Grouper

Overview

An identity and access management program based on InCommon Trusted Access Platform deploys Grouper as a strategic component of its institutional role and access management solution. Grouper is at the center of all group and access policy management. Managing access with Grouper results in access to target systems being automatically kept in sync with policy as subject attributes change in underlying systems of record (e.g. ERP, SIS, etc). This overall mechanism coupled with powerful distributed management capabilities is what makes Grouper a core component of the InCommon Trusted Access Platform.

The Grouper project maintains three introductory videos that are a bit dated, but still very relevant. The first one, Intro to Grouper: Access Management & Grouper , provides project background and the rationale for the project's approach to access management. The second in the series, Intro to Grouper: Grouper's Core Access Management Capabilities , explores specific Grouper concepts and capabilities, and how they come together in a specific case for managing access to a VPN service. The third, and final in the series, Intro to Grouper: Grouper Toolkit Components , describes the various product components and capabilities, and options for integrating with existing campus IAM architecture.

The University of Chicago VPN example described in the Intro to Grouper series, provides a great overview of how a variety of Grouper's capabilities come together to implement powerful access control management, and illustrates a common pattern that can be applied in many situations. The four basic steps are:

1. Leverage institutional data to create meaningful cohorts
2. Enable distributed management of policy exceptions and ad-hoc groups
3. Use composite groups to define access policy
4. Reflect access control decisions to target systems

Let's consider the access policy:

> ***"Staff, student, postdocs, and members of the IRB office are authorized to use the VPN unless their account is in the process of being closed (closure) or has been administratively locked by the Information Security Office."***

This is what NIST SP 800-162 calls the "natural language policy" (**NLP**).

Figure 1 shows how the NLP is translated into digital policy (**DP**) in Grouper.

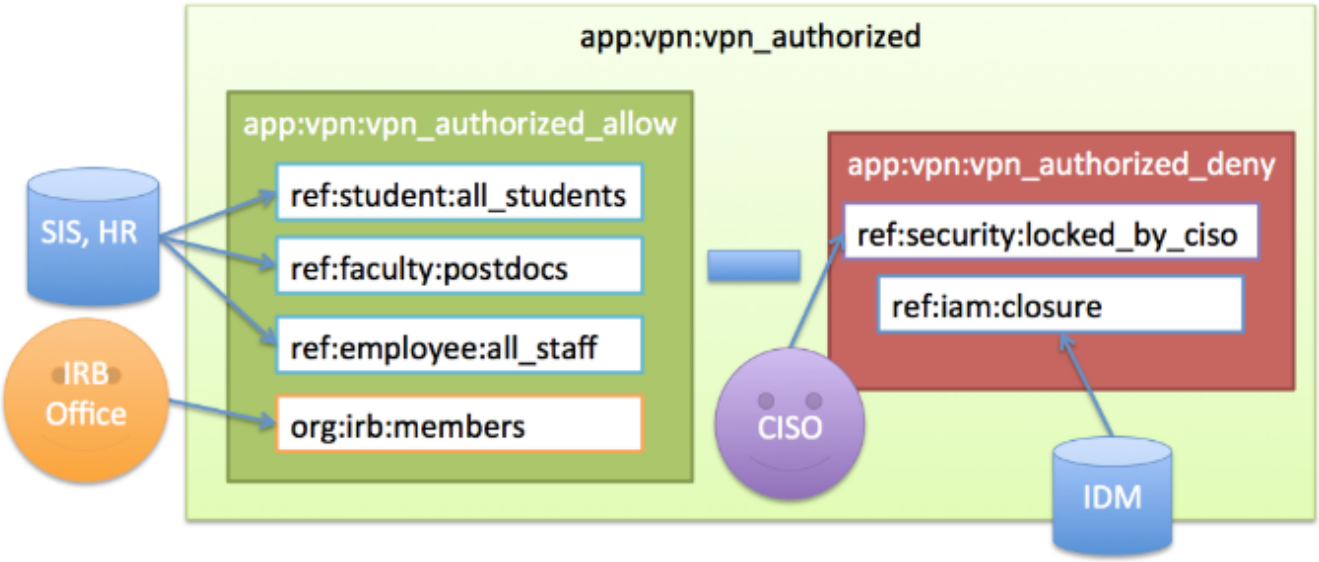


Figure 1: University of Chicago VPN Access Policy

The policy calls out a number of different cohorts which we call reference groups. These are groups of subjects that share some characteristics, such as being a student, a postdoc, or a member of the IRB office. Reference groups can be kept in sync automatically with institution data or manually when a data source is not available. The IRB office reference group is kept up to date by directly adding or removing members via the Grouper UI. Reference groups are institutional meaningful concepts and represent the best known "truth" about a subject at any given moment.

In the example above:

- Once the required **reference groups** are available, an **access policy group** `app:vpn:vpn_authorized` is created and configured to reflect the NLP.
- An **allow group** `app:vpn:vpn_authorized_allow` includes reference groups `ref:student:all_students` , `ref:faculty:postdocs` , and `ref:employee:all_staff` . This captures the first part of the NLP.
- Additionally, a **deny group** `app:vpn:vpn_authorized_deny` is created and includes an **identity lifecycle group** representing a deprovisioning state, `ref:iam:closure`, and a security control group `ref:security:locked_by_ciso`
- Combining the allow and the deny group in the composite group `vpn_authorized` yields the appropriate digital policy and the composite group is kept up to date as the underlying reference groups change.

Converting natural language policy into executable digital policy with a combination of reference groups and access policy groups is a fundamental Grouper pattern and objective. Grouper provides a single point of management, enables groups to be defined once and reused across multiple applications, and empowers the right people to manage access. This example also demonstrates a key objective of any Grouper deployment, which is that access policy should be easily discoverable and verified.

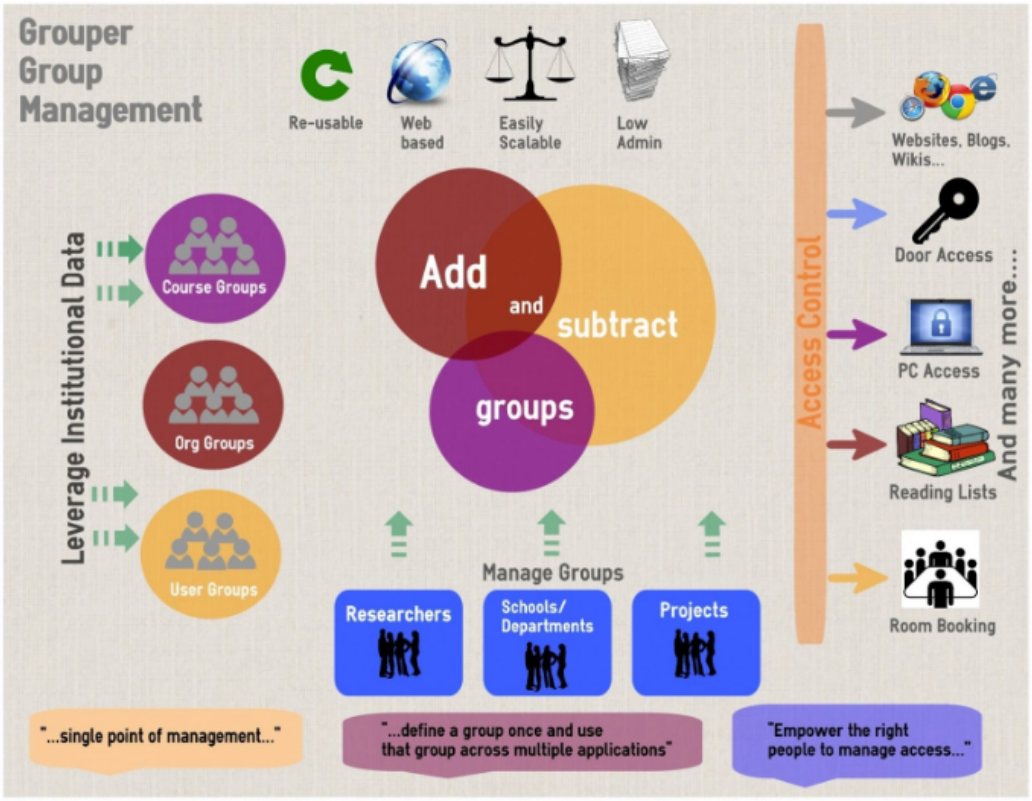


Figure 2: Newcastle University Grouper Infographic

The rest of this page will introduce core Grouper concepts and primitives which includes:

- Folders, Groups, and Membership
- Composite Groups
- Grouper Privileges
- Grouper Daemon/Loader Jobs

Folders, Groups, and Membership

Grouper is organized around three main concepts; **folders, groups, and memberships**. A folder is a container for other folders, groups, and other objects. It provides a namespace and a security context for the objects it contains. A group is a list of entities (other groups or subjects) that have membership in the group, along with other attributes that define the group, such as group name and description.

Membership in group can be direct or indirect and describes a relationship between a subject or group and the group of interest. A subject or group is a direct member of a group if the subject or group has been added to the group's membership list. A subject is an indirect member of a group, if the group contains a subgroup for which the subject is member, or as the result of a composite group. Any membership that is not direct is called indirect. All indirect memberships are automatically updated as the underlying direct memberships change.

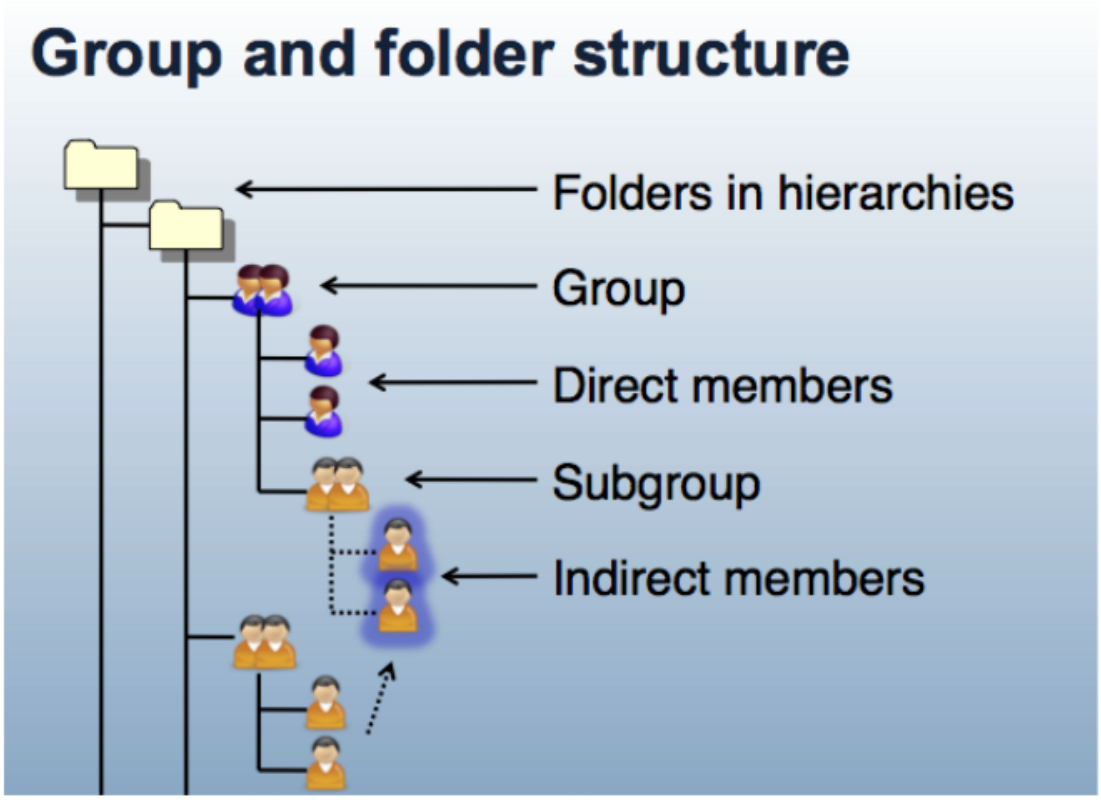
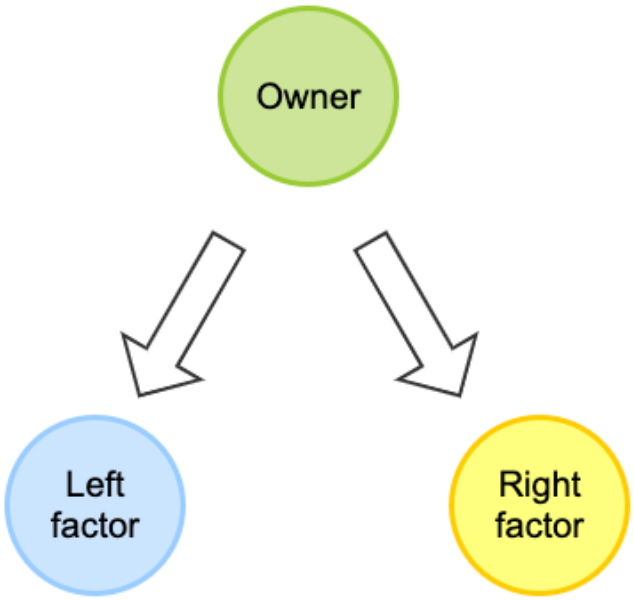


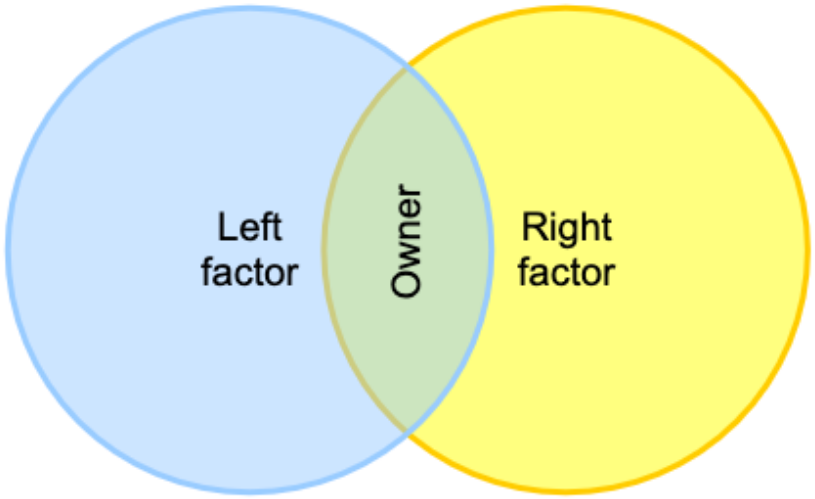
Figure 3: Group and Folder Structure

Composite Groups

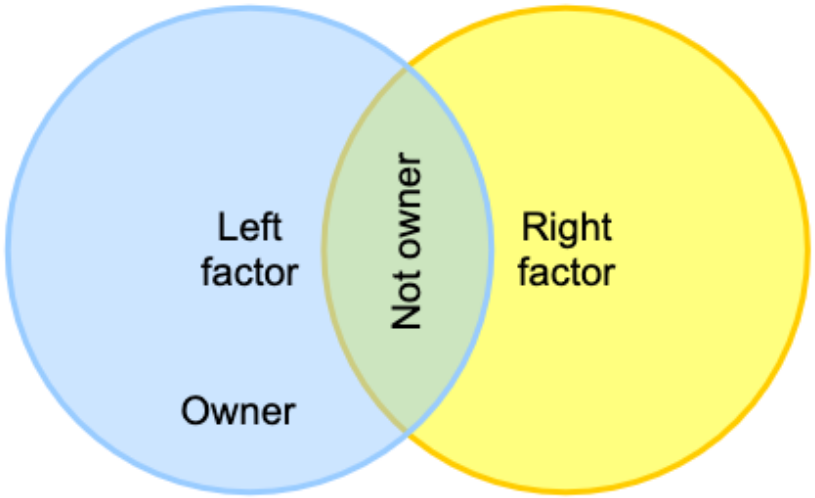
Grouper allows you to use two existing groups to define a third group. The third group is a **composite** of the other two factor groups. Groups can be combined as an **complement** or **intersection**



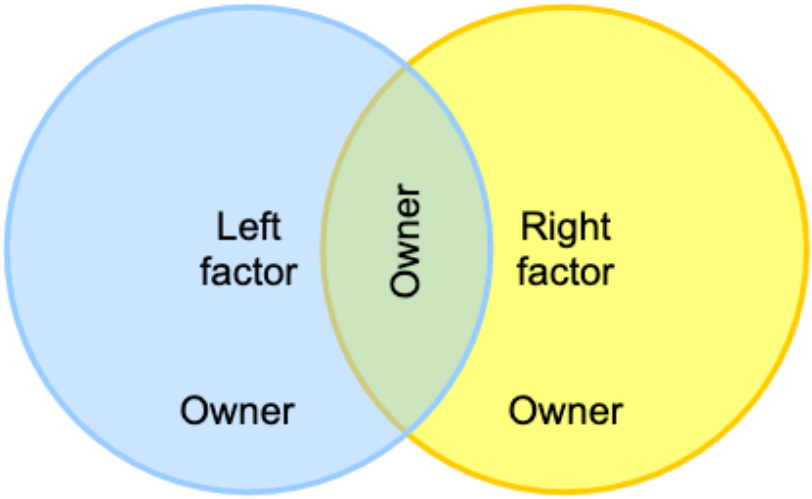
- **Intersection** includes entities that belong to both of the original factor groups, and produces a composite "members-in-common". Intersection groups are often used when creating reference groups from basis groups.



- **Complement** includes subjects that belong to the primary “left” factor group who are not also members of the secondary “right” factor group (i.e. “left” minus “right”). Complements are the top-level strategy for access policy. The “left” group representing an **allow group**, and the “right” group representing a **deny group**.



- Union is all members of the factors and is not a composite option since it is the same as adding the left and right factor as members of the owner



Here are two examples of using **intersections**:

- One might intersect the "employees" group with the "MFA enabled" group to yield “employees who have MFA enabled”. This composite could then be used in access policy that requires employees to have MFA enabled.
- Another common use is to intersect an ad hoc group with an "active" group. For example for an ad hoc group where members must be employees.

As membership changes in the factor groups they are automatically reflected in overall composite group.

Note you can implement multiple composite groups to make complex group math expressions. For example you can intersect three groups by using an intermediate composite group and an overall composite group.

You can also use Grouper Rules to model composite groups. A rule can detect actions, check conditions, and do resulting operations. So in the above example, an ad hoc group could have a rule that detects if a user of the group is removed from the employee group, and if so, then that membership in the ad hoc group could be removed or assigned an expiration date a certain number of days in the future.

Or the rule could email the group owners to review the membership. You could use the rule if you want the membership instantly and permanently removed. On the other hand, the composite would cause the non employee to be in the ad hoc group if they get re-hired. Grouper Reports and Grouper Deprovisioning can help with removing ineligible ad hoc members.

Grouper Privileges

Grouper Privileges control who can take what action on Grouper objects (i.e. folders, groups, and attributes). Grouper Privileges can be assigned to subjects or groups within Grouper. Each folder, group, and attribute has its own privilege assignments which enables fine-grained access control and delegation of authority. The Access Privileges definition in the Grouper glossary provides further details on what each privilege provides.

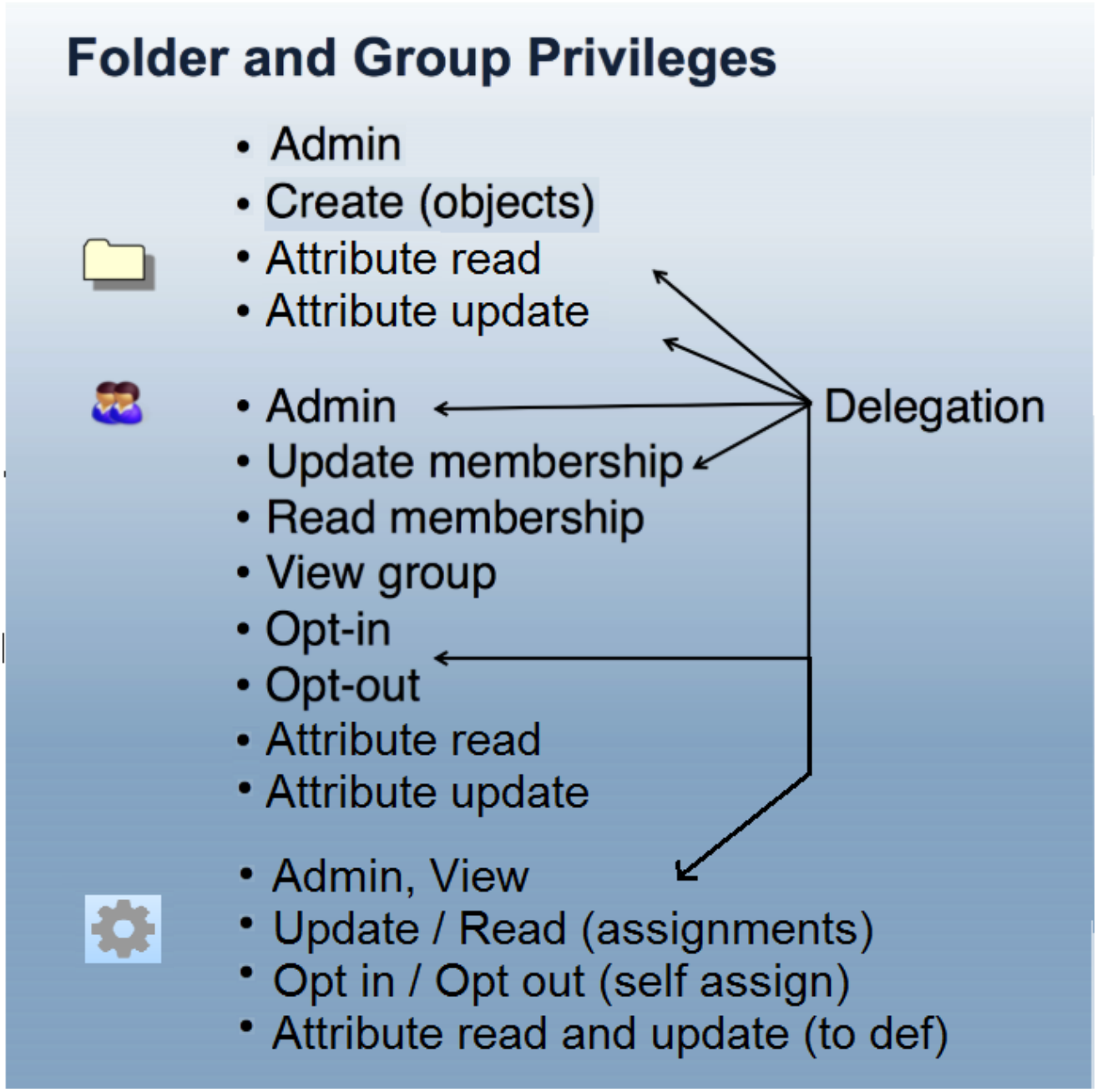


Figure 4: Grouper Privileges and Delegation

The **combination of folder hierarchy, security groups, and Grouper Rules** are used to manage privileges. Folders can be configured so that group s, attributes and folders created within a parent folder inherit privileges from the parent folder. How to design groups provides examples of setting up folder structures and configuring privileges. Grouper rules privileges inheritance on UI provides details on managing inherited privileges in the Grouper UI.

Grouper Daemon, Loader Jobs, and Data Flows

The Grouper daemon is a background process required for a number of key Grouper features including the Grouper loader. The Grouper Loader allows you to automatically manage group memberships based on a data source. Supported data sources include SQL and LDAP. Details about the various types of loader jobs and examples are maintained in the Grouper loader wiki page. Grouper Training Admin Loader Part 1 and Grouper Training Admin Loader Part 2 training videos also go into more details about loader job options and configuration, and operation.

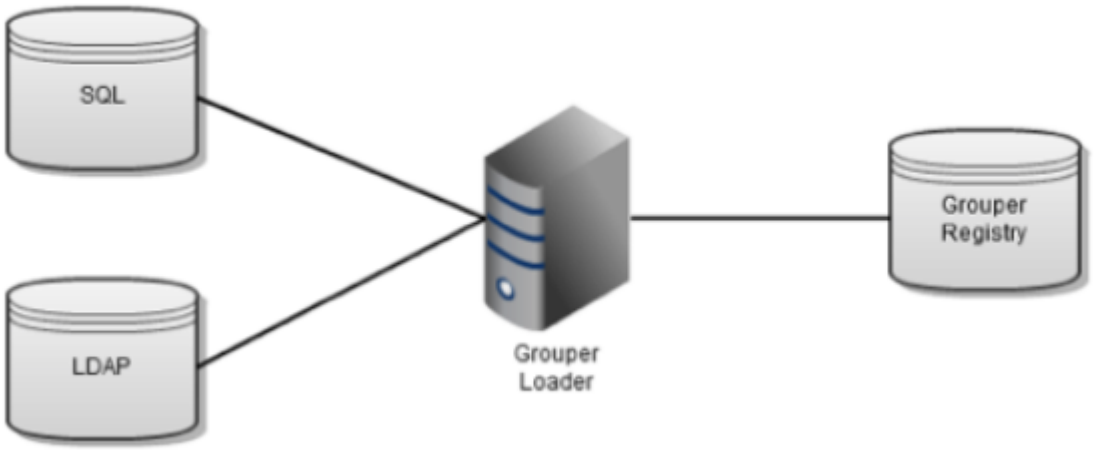


Figure 5: Grouper Loader Jobs

Data flow in and out of Grouper

Name	Description	In to Grouper	Out of Grouper
User interface (UI)	Manually manage groups, import / export CSV	Yes	Yes
Web services (WS)	Securely read and write Grouper resources	Yes	Yes
Grouper client	WS client library in Java or command line	Yes	Yes
Grouper loader	Load group memberships from SQL or LDAP	Yes	No
PSPNG	Provision memberships to LDAP/AD	No	Yes
SAML entitlements	Identity provider can read Grouper memberships (from LDAP or SQL or WS) and send these as entitlements to service providers during authentications	No	Yes
Messaging	Read or write messages with built-in messaging, AWS SQS, RabbitMQ, or ActiveMQ	Yes	Yes
Targeted provisioner	Provision to Box, Azure / O365, Google apps, Duo, Atlassian, etc	Not generally	Yes
SQL	Read Grouper data or provision SQL for an application	No	Yes
GrouperShell (GSH)	Command line interface	Yes	Not generally
Grouper Java API	Write Java using Grouper open source libraries	Not common	Not common
Change log consumer	Implement a Java interface to process Grouper events (near real time)	No	Yes
"Other job"	Implement a Java interface to have a scheduled task	Yes	Yes

Previous: Introduction to the GDG

Next: Folder and Group Design

Folder and Group Design

Overview

> *"Just having a plan or standard has been quite helpful, as it allows implementers to get on with real work without having to stumble on how to name things or where to stick them."*
> *- Tom Barton, University of Chicago*

Once Grouper is initially deployed it is up to the Identity and Access Management (IAM) analyst to construct and organize the appropriate folders and groups necessary to achieve the desired access management capabilities. The folder and group design provides institutional-level and application-specific group definition and management, and supports the campus-wide scope of the service. Such a plan enables organized service growth and promotes effective reuse of common objects.

This section first defines a variety of group types and purposes and then describes a recommended initial folder and group organization.

An efficient way to set up groups that adhere to the recommended structure is using the Grouper Template Wizard.

Group Definitions

Basis Groups

Often the best source of data for building meaningful cohorts is a combination of arcane codes representing various types and states of employees or students, and often sourced from multiple systems. To leverage the power of Grouper, these groups should be brought in as “raw” **basis groups**.

Basis groups are used by the IAM analyst to construct the cohorts that are required for access policy. Access policy does not use basis groups directly, rather the basis groups are used to build up reference groups. This indirection provides the IAM analyst the ability to adjust to changing source systems and business practices while keeping reference groups and access policy relatively stable. Basis groups are typically only visible to the IAM analyst, and would not normally be reflected out to applications and directories.

Reference Groups

Reference groups tend to be organized in particular folder locations for convenience and ease of use, but what makes a group a reference group is not its name or folder location, but rather its intended use, definition and scope, and data management expectations.

A **reference group** is a set of subjects that is largely intended to be used by reference within access policy. **Reference groups can be thought of as labels or tags that identify meaningful cohorts**. In this way, they can also be viewed as subject attributes from an ABAC perspective. Access policies often require cohorts organized via institutional affiliation (faculty, staff, student), a particular office or department (president’s office, finance division, chaplain), program (chemistry students), and even residence or class year. All of these are good examples of reference groups.

Reference groups represent the best possible source of “truth” about any particular subject at a given time for the purposes of access control. Therefore, the rules that define the various cohorts must be well understood and known. Reference groups may have institutional scope (e.g. student, faculty, staff) where the definition is expected to apply globally. Reference groups can also have application or organizational scope in which case the definition only applies to limited set of applications or policy definitions.

Reference groups are intended to support effective and efficient day-to-day operations by providing timely, accurate groups representing various cohorts required for access control and collaboration. Data for placing subjects into a particular cohort is often available in source systems or operational data stores. However, in cases where a source system is not available an authoritative office may be responsible for maintaining membership directly via the Grouper UI.

Ideally, manually managed reference groups should only be for small cohorts that lack sufficient institutional data. If you find yourself manually managing large reference groups, look for good sources of data for a loader job or other basis groups. Sources of data, timeliness of updates, reliability, and administrative access control are expected to be well known since they will directly affect access to a wide variety of services and resources.

When viewing a group in the Grouper UI, under the “More tab” click “this group’s membership in other groups” to show where the reference group is used in access policy.

Access Policy Groups

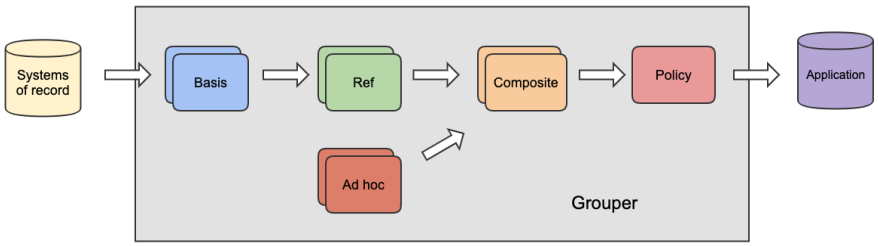
Access to services is often controlled by membership in a particular group or by having a certain subject attribute value (e.g. eduPersonEntitlement). This is sometimes called coarse-graining access control to distinguish it from the fine-grained permission management that is often found in RBAC like systems. Access policy groups can be used to deny or allow access to a resource (e.g. access to VPN) based on policy or to place a subject in an application specific role.

ABAC natural language policy, that is access policy stated in common language, must be converted to digital policy for any access control mechanism to effectively operate. Digital policy is manifest in Grouper via access policy groups. Subject membership in an access policy group must be indirect and represents a precomputed access policy decision based on subject attributes (i.e. the subject's membership in various reference groups).

An **access policy group** is a composite group whose membership is composed of an **include group** (i.e. the **allow** group) and an **exclude group** (i.e. the **deny** group). Subject membership in both the allow group and the deny group should be indirect (i.e. through reference groups) and have a clear mapping to the natural language policy. When exceptions to policy are necessary, locally scoped reference groups should be added.

It is good practice to **limit access policy groups to indirect membership assignments via reference groups**. This ensures that as subject attributes change, effective membership are kept up to date and access control decisions are correct. It also enables the direct mapping from natural language policy to digital policy and vice versa. Individual exceptions to policy, while not expressly recommended, can be accommodated by adding subjects directly to the allow/deny groups.

Membership in an access policy group is often kept in sync directly with a target service or an intermediary like an LDAP-based enterprise directory service. Services can also query Grouper directly for membership assignment.



Account Policy Groups

Services almost always require some type of identity record to be maintained at the service as a first step in granting access. This could be as simple as a single identifier and a few subject attributes. Membership within an account policy group signals that a suitable identity record (i.e. an account) should be created and kept in sync at the target service.

An **account policy group** is a composite group whose membership is composed of a **single include group** (i.e. the allow group) and a **single exclude group** (i.e. the deny group). Effective membership in an account policy group represents a precomputed account policy decision.

Manual Groups

Groups designated as manual indicate that they are intended to be managed by a person. Examples of manual groups are:

1. Teams that cannot be derived from a query to another system
2. Includes or excludes for exceptions to a policy driven by automatically loaded groups
3. Fine grained assignments that do not follow a broad collaboration group structure

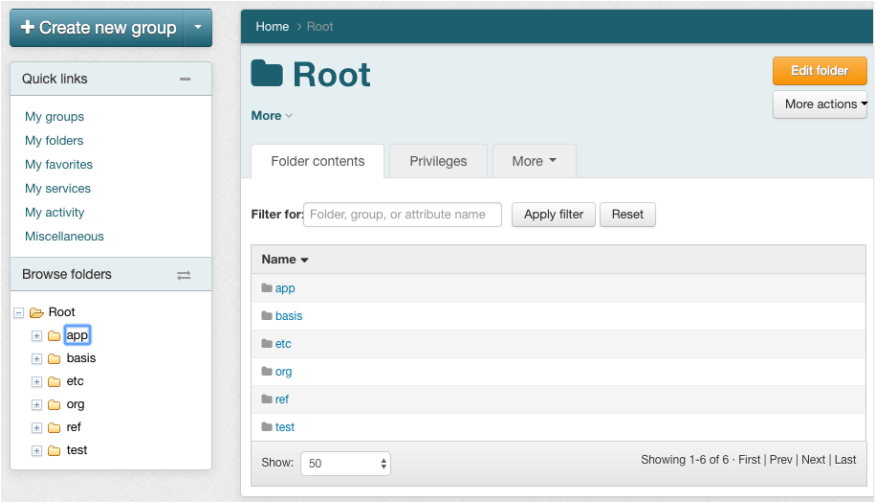
Intermediate Groups

Groups that do not fall in to another category might be "intermediate" groups. These are considered internal, or they exist to make a policy work but the average Grouper user does not need to be concerned with them. While building a policy out of composites, you might have some groups that are marked as "intermediate" to indicate that they are necessary but can be generally ignored.

Standard Folder Set and Pattern

The recommended standard folder set and pattern consists of **six folders** that are generally organized by group type, intended use, and visibility.

- `etc:` - Grouper configuration, administrative access control groups, and loader jobs
- `basis:` - groups used exclusively by the IAM team to build reference groups
- `ref`: - reference groups (i.e. institutional meaningful cohorts) - "truth" about subjects
- `app:` - enterprise applications access control policies - specific policies for services
- `org:` - delegated authority, ad-hoc groups, org "owned" apps or reference groups
- `test:` - test folder for system verification, and group math sandbox



A main reason for a folder structure if to delegate access to sub-organizations and teams. Grouper privileges are inherited from folders. Also some groups belong together but it is unclear if they are a certain type or another. For example some people might consider a group to be a basis group, and others might think it is a reference group. "Institutional meaning" can be vague. Therefore you will have conflicting requirements when trying to follow the folder structure but also organize the groups so that they make sense and have inherited privileges. In Grouper you can identify folders or groups as a certain type so that groups can have a type but live in a non-standard location. i.e. most of your reference groups might be in the "ref" folder, but you will also have reference groups peppered throughout your Grouper registry. Pay attention when assigning types to folders since that will inherit to all groups in the folder. For example, a "policy" folder in an app might container non-policy intermediate groups. For this reason you should avoid assigning the "policy" type to a folder.

etc folder

The `etc:` folder holds various Grouper system specific configuration, loader jobs, and the Grouper system access control groups.

- `etc:grouper_ui` - members can login and use the Grouper User Interface
- `etc:grouper_ws` - members can call and use the Grouper Web Services
- `etc:grouper_admin` - members have root-like privileges to the Grouper system
- `etc:loader:` - folder for various loader jobs

> Tip: Initializing administration of privileges provides further details on setting up and configuring Grouper system access control groups.

basis folder

The `basis:` folder is strictly for use by the IAM team and provides a place to pull in and manage “raw” basis groups. **Basis groups are used to build up institutionally meaningful reference groups**. Basis groups provide a layer of abstraction between reference groups and source systems, and shields access policy from low level details of any particular source system. This layer provides the IAM analyst the flexibility to build appropriate reference groups and keep access policy groups focused on institutionally meaningful concepts. The folder structure under `basis:` should help the IAM analyst understand, find, and manage various basis groups. This folder might be further organized by source system.

`basis:hris:{employee_codes}` - types of employees, used to build reference groups

`basis:sis:{student_codes}` - types of students, used to build reference groups

ref folder

The `ref:` folder is for institutionally meaningful reference groups. These may be built from basis groups, inline loader jobs, or manual maintained with the Grouper UI. Reference groups are intended to be used in access policy and help provide a direct mapping to the natural language policy. The folder structure under ref: should help the access policy manager understand and find the appropriate reference groups. Generally, sub folders are used to organize various kinds of cohorts.

`ref:role:` - institutional scope roles (e.g. president, provost, chaplain...)

`ref:employee:` - types of employees (faculty, staff, part-time, full-time...)

`ref:student:` - types of students (class year, on-track-grad, incoming-class...)

`ref:alum:` - types of alumni

`ref:course:` - course rosters including instructors, TAs, etc

`ref:dept:` - organization hierarchies

app folder

The `app:` folder is used to organize the groups and folders required to effectively manage access policy for a particular application. This may come in the form of access policy groups, account groups, and application specific reference groups. Each application has its own folder and a similar sub folder structure. Grouper templates can be used to build the application folder structure and to add a policy group.

`app:foo:` - root folder for the “foo” application

`app:foo:security:` - folder for administrative security groups for this application folder and group set

`app:foo:security:fooAdmin` - members have root-like privileges for the app:foo: folder and group set

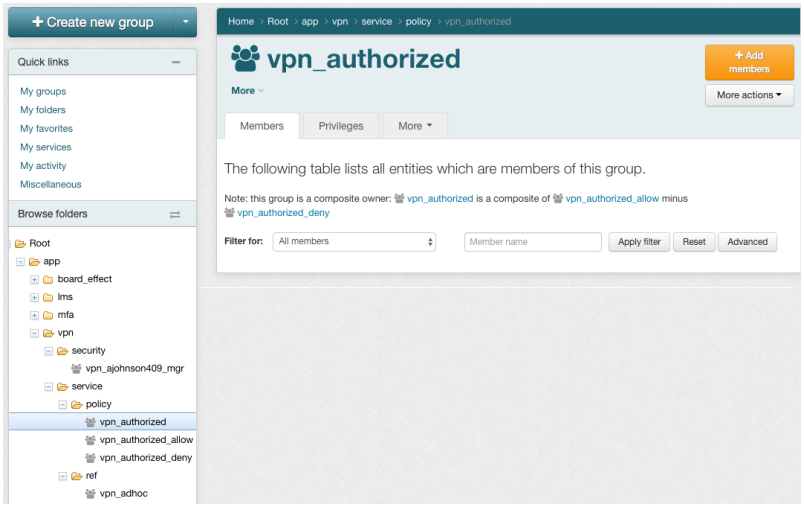
`app:foo:service:ref:` - folder for “foo” application specific reference group if needed

`app:foo:service:policy:foo_user` - access policy group, composite group (foo_users_allow - foo_users_deny)

`app:foo:service:policy:foo_user_allow` - generally this contains reference groups, composites, or manual groups. This is an 'intermediate' group type.

`app:foo:service:policy:foo_user_allow_manual` - only direct members people who should be included in the policy as exceptions, and are not automatically included. This is a 'manual' group type

`app:foo:service:policy:foo_user_deny` - may include `ref:iam:global_deny`. This is an 'intermediate' group type.



org folder

The top-level folder structure and overall Grouper deployment is typically managed by IAM architects within a central team IT department. This can be a bottleneck to adoption in large institutions. The `org:` folder provides a namespace for distributed IAM management. A distributed IT organization would be given root-like administrative privileges on a particular org: folder and could managed the namespace independently without the help of central IT. The org: folder is also sometimes used to delineate ownership of applications, and may be used for organizational scoped or maintained reference groups. The folder structure may replicate the top-level folder structure, but will be scoped to the particular organization.

`org:compsci:etc:compsci_admin` - members have root-like access to `org:compsci:`

`org:compsci:ref:` - Computer Sciences Department managed reference groups

`org:compsci:app:` - Computer Sciences Department applications

test folder

The `test:` folder is generally used by the IAM analyst for isolated verification testing of new production provisioning components and as a sandbox for group analysis and exploration.

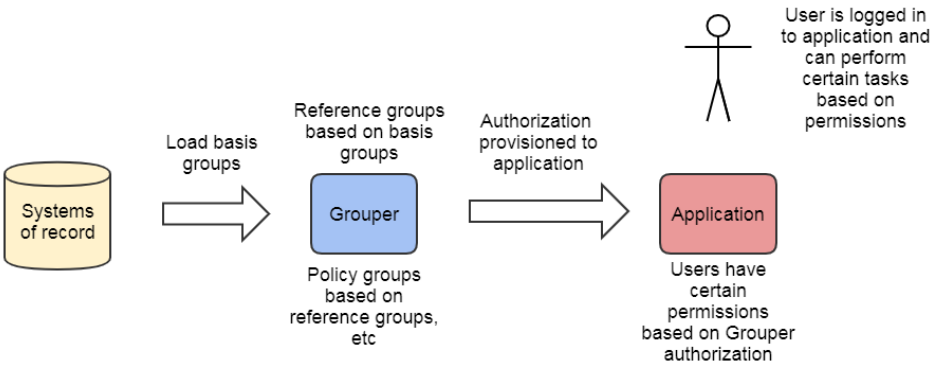
Previous: Understanding Grouper

Next: Access Control Models

Access Control Models

Overview

The general Grouper approach to access governance is to create and maintain access policy groups which are built from institutional meaningful cohorts (reference groups), ad hoc exception groups, and indirect basis groups. Membership in the access policy group represents a pre-computed access policy decision. The policy decision is typically communicated to the target services via LDAP or SAML, but can be done in a variety of ways. To help implement these access control models, review the documentation on the Grouper Template Wizard.



The target service uses the policy decision from Grouper to determine what level of access the user has. The policy decision could be mapped to a hard-coded application role or a fine-grained permission set managed locally in the service. For example, the user might be dynamically assigned to a group in Confluence due to a SAML entitlement provisioned via Grouper. The user might then have access to projects due to configured access in the Confluence admin screens. This is explained in the "Policy groups and dynamic application permissions" model below.

Mechanisms to communicate and enforce policy decisions can vary considerably depending on the security needs and capabilities of the target service. However, the overall approach to access governance with Grouper remains consistent. The rest of this section uses terminology and models from NIST SP 800-162 and XACML to demonstrate a variety of models leveraging the overall approach. Note, you could use multiple models at once, or could engineer hybrid approaches.

Access Control Models

Most of the **Access Control Models (ACM)** lend themselves to distributed access control, meaning the authority to manage an access control policy or exceptions to policy can be delegated to authorized people. The most common ACMs are listed here, with a diagram and more details below on each one.

Access Control Model	Type (s)	Description	When to use	Notes
Policy groups and static application permissions	Policy groups	Roles are managed in Grouper and the permissions of those roles are hard-coded in the application or are opaque. This is a very common access control model.	If the application has hard-coded permissions based on application roles, and the roles can be provisioned from Grouper	See the U. of Chicago VPN access diagram /example and the examples on the Example Access Policies page.
Policy groups and dynamic application permissions	Policy groups	Very similar from a Grouper perspective to "Policy groups and static application permissions". Authorization configuration is in two places, who has which role (in Grouper), and what each role/user can do (in app)	If the application can configure permissions based on users/roles, and roles can be provisioned from Grouper	Two users in the same role might have different permissions if there are individual permission assignments in addition to role permissions assignments. See examples for Box and Duo.
Policy group for coarse-grained access	Policy group Coarse grained authn	Identify the population of who should be able to log in to the application, make a policy group, and lock out users not in that group	Use this for local or SaaS applications. This drastically improves your security posture. Use this if you cannot integrate roles with the application, or even if you can!	This can be setup as a reverse proxy to protect the application from any unauthenticated, unauthorized access. See the U. Penn PeopleSoft example.
Grouper for access reporting	Access reporting	Access is configured in the application. Assignments in the application are loaded into Grouper to help with reporting and deprovisioning	Use this if the roles/permissions in the application cannot be externalized to Grouper. Its very valuable for Grouper to track the assignments.	You can use Grouper attestation, Grouper deprovisioning, etc. on read-only access assignments

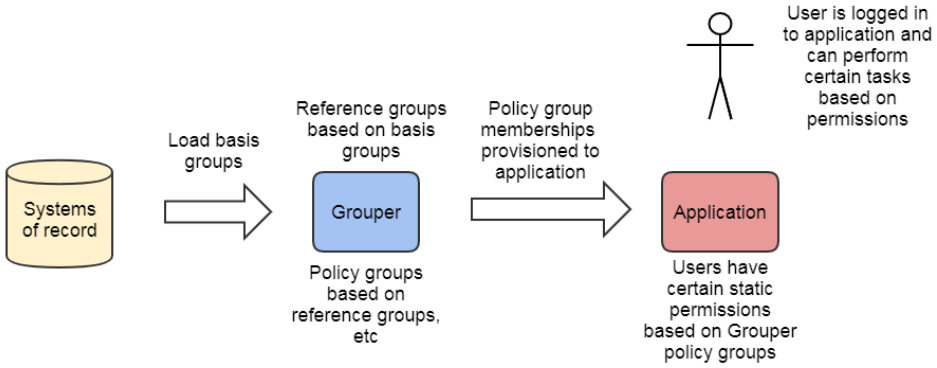
Grouper managed permissions	Policy groups Externalized permissions	Policy groups are used as roles in Grouper and permissions are assigned in Grouper to the roles /users. Permissions are provisioned to the application	For custom application where you want to offload the permissions to Grouper as an RBAC engine. Its possible though less common for packages as well.	The difficulty can be if the app can support externalize permissions, if the grouper RBAC permission model fits the application, and if the generic Grouper permission screens are usable for the application
eduPerson Affiliation for authorization	Reference groups	Use eduPersonAffiliation or equivalent to secure access e.g. based on if the user is a student or employee	Use this if the application (SaaS?) only supports security by eduPersonAffiliation	Less mature, less flexibility, not recommended Use only as a last resort You could work around this it might not be ideal Lacking delegation

The following sections provide a diagram and more detail on each of the ACMs listed in the table above.

ACM Policy groups and static application permissions

Like many ACMs that use policy groups, access policy administration and the policy decision point (pre-computed membership assignments) are done in Grouper and can be communicated to the target service in a variety of ways. Access policy groups in Grouper enable direct mapping from natural language policy to digital policy and back, and policy is kept up to date automatically as subject attributes change. Grouper policy groups support audit, compliance, and attestation. This model can be broadly used with a variety of services.

1. Access policy groups configured in Grouper based on institutional meaningful cohorts (i.e. reference groups, ad hoc groups, etc)
2. Authorization provisioned to application via LDAP, SAML, web services, direct provisioning, etc
3. Application uses those role(s) with hard-coded or static permissions to allow the user to perform actions

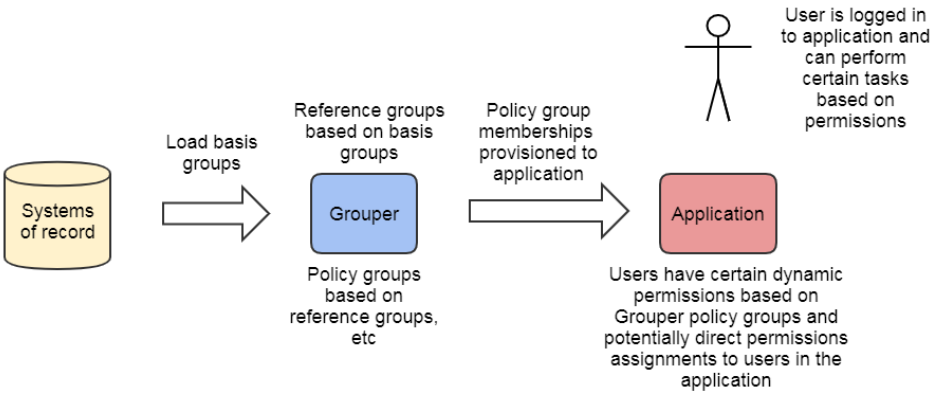


ACM Policy groups and dynamic application permissions

In applications with sophisticated RBAC capabilities, fine-grained permission sets are typically configured via an administrative interface within the application itself. These permission sets are then associated with a role that can be mapped to a set of users. In this model, the user to role mapping is done in Grouper by pairing a normal access policy group with the role defined at the target service. The policy of which subjects are mapped to application roles is similar to other ACMs with policy groups.

1. Access policy groups configured in Grouper based on institutional meaningful cohorts (i.e. reference groups, ad hoc groups, etc) provides User -> Role mapping
2. Authorization provisioned to application e.g. with LDAP, SAML, WS, direct provisioning, etc
3. Fine-grained permission sets are managed at the target service (Role -> Permissions and User Permissions)

If permissions are also assigned to individual users using the the application's permission management interface (not necessary for this ACM), then Grouper's view of what a user has is not necessarily the same as the application's. Two users who have the same role in Grouper could have different permissions in the application. Grouper knows at a high level what access the user has, but you would need to consult the application for the full picture.



ACM Policy group for front door access

Often for pragmatic reasons one wants to limit access to a target service by preventing authentication from completing or by preventing network traffic from reaching the application. This "front door" access control provides a logical outer perimeter in addition to the applications normal authorization mechanisms. This could be used in situations where the target application does not have sufficient access controls either due to technical or administrative reasons, or to provide another security layer.

In cases where it is preferable to completely limit unauthorized network traffic from reaching the application, a reverse http proxy can be used block unauthorized and possibly nefarious network traffic. Combing "front door" access with other ACM models enables defense-in-depth. Several cases where this is necessary or desirable are; 1) when the target service has insufficient access controls to limit access based on the desired policy, 2) when the target service lacks a good unauthorized user experience, and 3) when the application runs on software that might have 0-day exploits and the security patches are difficult to keep up with.

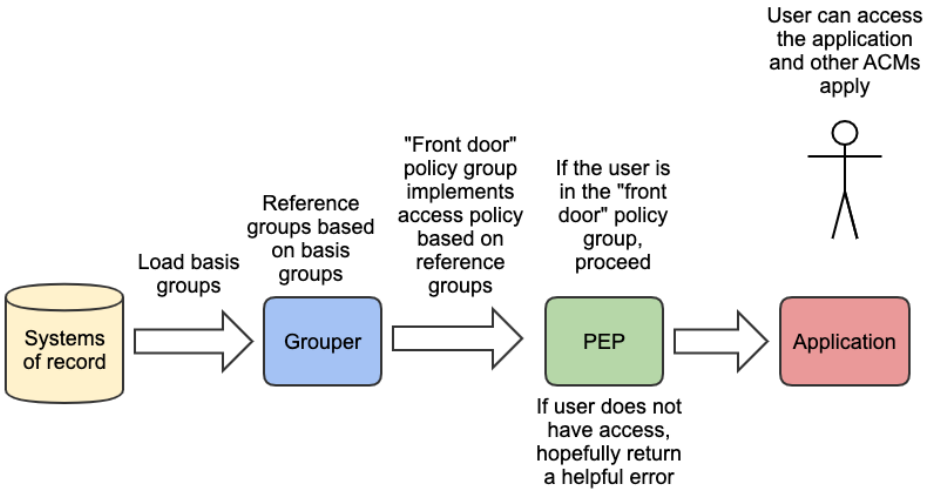
The "front door" policy group membership could be wider than what is actually authorized to use the target application, if the application additionally provides its own authorization (e.g. identify the admin users). So at a minimum the policy group could contain an affiliation (e.g. "employee" reference group), even if only certain employees have access to the application. The most precise "front door" policy group would contain only the subjects that have access to the application, either via Grouper policy groups or application specific configuration.

There are several options for the "front door" ACM **policy enforcement point (PEP).**

PEP	Description	When to user	Notes
SAML Identity Provider	The Shibboleth IdP can block access or send a blank assertion if the authenticated user is not in a certain Grouper policy group when authenticating to a certain Service Provider (SP)	SaaS services Any SP integrated with your IdP	There might or might not be a friendly error page Some IdP operators might not to mix authentication with authorization on principal Some SaaS apps might allow SAML authentication in addition to local "back door" authentication, so it might not provide the sufficient intended protection in those cases (e.g. box) You could mine IdP logs to see what population has recently used an application to make sure you are not making the front door policy group too restrictive
Load balancer (e.g. F5)	Application load balancers that reverse proxy http traffic might be able to limit traffic by a subject attribute or group membership. For example, the F5 load balancer has been successfully integrated with SAML, and can restrict traffic to an application by entitlement from Grouper. This could be a security reverse proxy component as well, or a CASB (Cloud Access Security Broker).	Any application behind a load balancer that authenticates the user.	The application needs to be reverse proxiable The UI (SAML) traffic needs to be separate from any WS or public traffic Blocking all network traffic by a reverse proxy improves the threat model since attackers must be authenticated and authorized Instead of a load balancer, the component could be an application firewall
Web server (e.g. Apache)	Web servers can reverse proxy traffic or just serve pages and can block all network traffic unless someone is authenticated and authorized	Any application reverse proxied behind an apache Any apache application	The application needs to be reverse proxiable if using apache as reverse proxy The UI (SAML) traffic needs to be separate from WS or public traffic Blocking all network traffic by a reverse proxy improves the threat model since attackers must be authenticated and authorized

Service provider (SAML SP)	The SAML SP can block authentications if not in a certain Grouper group	Any application that uses a SAML SP	It's possible that nefarious traffic could attack the application since the SP is generally along side the application and not in front
"Can login" role	The application might have a group that indicates a user has access (e.g. jira-users)	If the application has a "can login" group	The provisioned list of accounts, or a role of "can login" or "user" is similar to a coarse-grained security control This does not help secure network traffic like a reverse proxy

1. A "front door" access policy group is configured in Grouper that indicates who can use an application
2. Authorization provisioned (e.g. LDAP, SAML, etc) to the enforcement point (e.g. IdP, F5, Apache)
3. If a user is not authorized they should be directed to the "not allowed for this app" page
4. The application should do its own security using another ACM (e.g. get roles from other Grouper policy groups)



ACM Grouper for access reporting

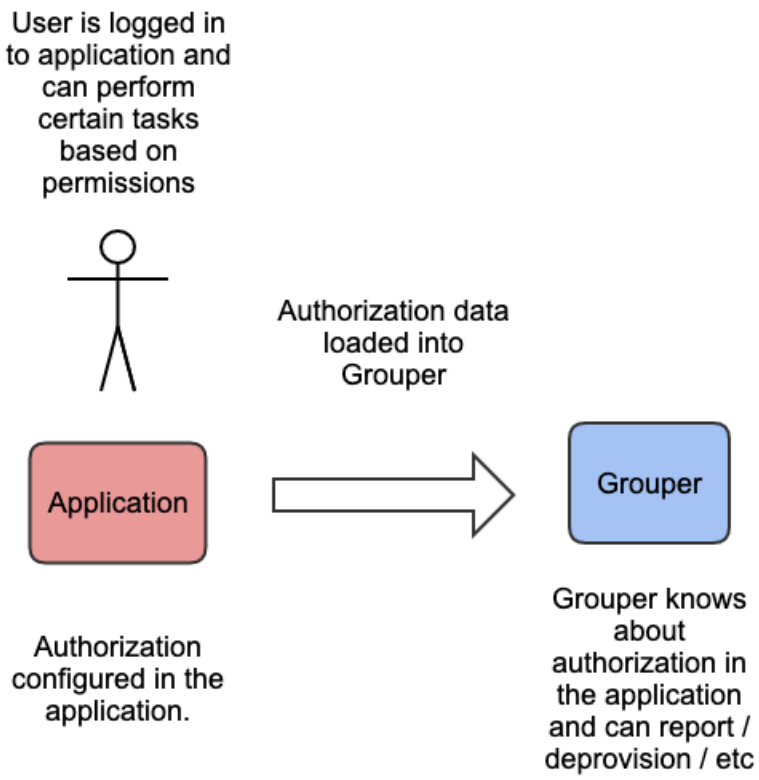
It is ideal if Grouper can be system of record for authorization for an application, but in many cases it is not possible. This could be because the application cannot use external roles, or because the application is preferred to manage those things internally (e.g. the UI in the application is needed for use). In this case hopefully you can feed authorizations from the application, or just who has any authorization, into Grouper. Now when you go to Grouper and see what someone has access to, you can see that application in the list.

Loading application authorization into Grouper needs to be a regularly schedule task so the data does not diverge. It could also be loaded near real-time. Several options exist for import authorization data:

1. Grouper loader via SQL or LDAP (preferred)
    a. You could ETL the data to a SQL or LDAP first
2. Web services
3. Custom job
4. Manual process (e.g. export the authorizations to CSV and import into Grouper. Do this periodically (e.g. monthly)

Once the authorizations are in Grouper you can use Grouper reporting, rules, attestation, deprovisioning, auditing, etc. This data in Grouper is readonly since it is sourced and managed in the application. In this case, deprovisioning or attestation must involve a user using the application to remove the unnecessary authorizations.

1. Authorizations are managed in the application
2. Authorizations are loaded into Grouper
3. Deprovision, attest, and report in Grouper by making security changes in the app

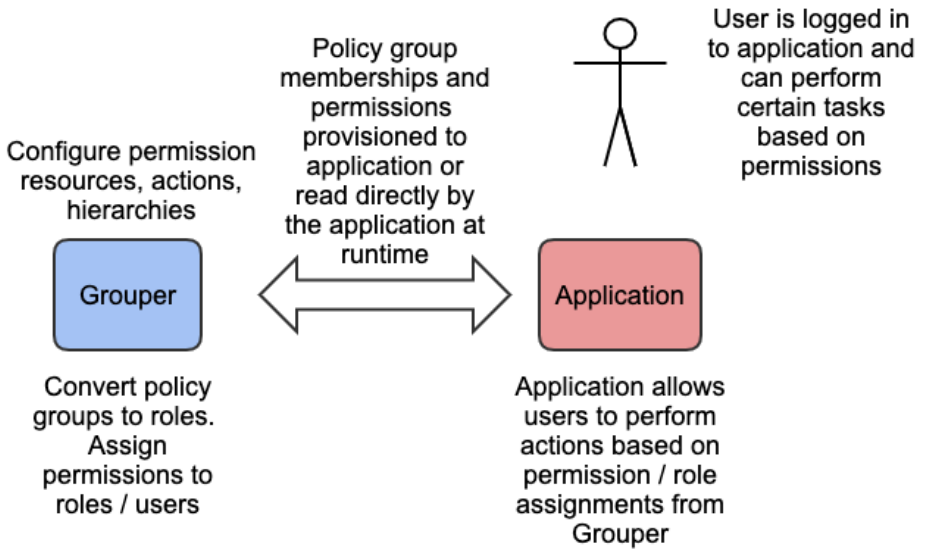


ACM Grouper managed permissions

While mostly out of scope for this guide, Grouper does have an advanced RBAC capability that is suitable for externalizing application permission management. In this model, the target system relies on Grouper for application permission management, including permission definition, role and resource hierarchies, and role to permission mapping.

Grouper role and permission management provides more details on this advanced use of Grouper.

1. Create policy groups similar to other ACMs
2. Convert those groups into roles
3. Configure permission resources and assign to roles / users (in Grouper)
4. Provision roles/permissions to application. Note, something more complex than LDAP/SAML is generally used, e.g. WS or SQL

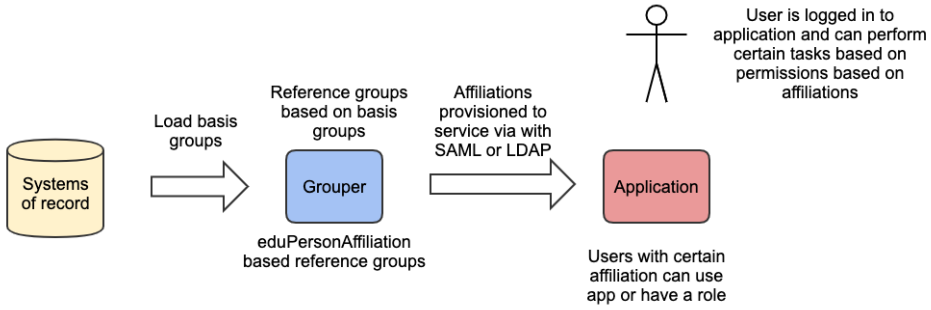


ACM eduPersonAffiliation for authorization

In this model, Grouper is used to master subject attributes that represent some type of affiliation or status at the institution. Generally these are reference groups for eduPersonAffiliation: member, student, employee, etc. Actual access policy administration is completely local to the target service. This model was more common years ago before sophisticated authorization was common. It is not recommended to do this since you are back to reference groups without the flexibility of policy groups.

This model is useful for cases when there is an informal relationship between the institution and the service provider, and a locally defined notion of the subject attribute like eduPersonAffiliation is sufficient for access control. However, the model breaks down quickly if a more exact notion of the subject attribute is required or if it needs to be different across services. It is important to remember that cohorts (affiliations, status, class year, etc) are not access policy. Do not be tempted to create service specific versions of reference groups. Favor creating access policy per service instead. Any application using this model should be comfortable with eduPersonAffiliation standard's notion of "broad-category affiliation assertions".

1. Subject attributes like eduPersonAffiliation are mastered in Grouper
2. Affiliations provisioned to the application typically via SAML Authentication Response
3. The service allows access or uses the affiliation to map to an application role / permission set



Previous: Folder and Group Design

Next: Provisioning Models

Provisioning Models

The access control models described in this guide all assume some mechanism to communicate Grouper group and membership changes to target services or an intermediary like an LDAP based enterprise directory service. Provisioning may be set up to keep various groups in sync with target systems, translate a group membership to an eduPersonEntitlement value, or create and keep remote identity records up to date.

Grouper provisioning mechanisms broadly fall into several categories:

1. "**Direct from Grouper to target service**" covers Grouper specific components and plugins for various targets such as AD/LDAP, Duo, etc. Grouper contains a change log and provisioning engine for loosely coupled connections to external systems.
    a. Grouper has built-in provisioners
    b. Interfaces could be implemented for custom provisioners (provisioning interface or change log consumer)
2. "**Message queue based delivery**" relies on a message queue infrastructure to communicate changes to appropriate provisioning components. In this model the logic for communicating with the external system would not be executed / managed / monitored / audited inside of Grouper
3. **External systems** can use web services, LDAP, or SAML to pull data from Grouper into their data repository.
4. **A non-Grouper provisioning engine** can get data from Grouper and provision it to the target system. Grouper can provision the middleware provisioning system (e.g. MidPoint) directly, or with a message queue, or it can pull from LDAP / SQL / WS.

Group and membership changes are provisioned to target services with two main strategies:

1. **Full-sync batch scheduled provisioning** looks at the source and the target and fully synchronizes the data
2. **Incremental near real-time provisioning** looks at the change log to send focused events to the target.

It is best to do both full and incremental provisioning if possible. The full and incremental sync should not run at the same time (they should wait until the other is done).

Whether you use one or the other, or both models, largely depends on your specific situation and provisioning targets. The Grouper Provisioning: Locally & Cloud slides from 2016 Technology Exchange provide more details on these approaches.

Previous: Access Control Models Next: Operational Considerations

Operational Considerations

Here are key places in the Grouper documentation and training with information on operational considerations:

- Tools & Topics for Ongoing Administration
- Ongoing Administration Tasks
- Grouper Training - Admin - Maintenance - Take 1
- Grouper high availability
- Grouper diagnostics
- Grouper report

You can monitor the health of the Grouper using the Grouper diagnostics URLs at http://{hostname}/grouper/status?diagnosticType=[trivial|db|all] for the Grouper UI, and http://{hostname}/grouperWS/status?diagnosticType=[trivial|db|all] for Grouper WS. If everything is ok, a 200 HTTP code will be returned, otherwise a 500 is returned with a description of the issue. The diagnostic URL has many options and is suitable for monitoring by systems like Nagios, Big Brother, etc. If you do not see the word SUCCESS on the “all” page, then something is wrong. Have monitoring tools like Nagios look for SUCCESS.

Use the Unresolvable Subject Deletion Utility (USDU) to clean up membership assignments for subjects that are no longer resolved by the Subject API.

Previous: Provisioning Models

Next: Conclusion

Conclusion

The Grouper Deployment Guide builds on industry standards such as NIST SP 800-162, and a synthesis of community practice to provide the terminology, concepts and strategies necessary to deploy and operate an effective InCommon Trusted Access Platform Grouper installation.

The overall strategy and access models described in this guide can be applied to a wide set of access governance requirements. However, the guide does not cover all possible uses of Grouper or even its complete feature set.

Be sure to review Grouper community contributions and Grouper use cases by category for additional examples of how to effectively use Grouper. Also, please share information on your institution's Grouper deployment to enhance this important library of user stories. See the community contributions wiki page for details on how to share your Grouper story.

This guide will naturally evolve along with community practice and as Grouper's capabilities improve.

Previous: Operational Considerations

Next: Example Access Policies

Example Access Policies

Example Access Policy 1 - Computing Lab | Example Access Policy 2 - Access to Online Course Material

This section explores several example access policies and how they might be implemented using the strategies described in this guide. They are provided to help reinforce how the concepts in this guide can be used to translate natural language policy into Grouper digital policy. The TIER provisioning state change progression models also provide useful examples.

As a general strategy, each of these examples uses a single access policy group which is itself a composite group of an allow group and deny group. The allow and deny groups may only include reference groups as direct members. These may be centrally managed under the "ref:" folder, maintained by an organization under the "org:" folder, or be application specific reference groups under the "app:" folder. Grouper has many features to automatically manage membership in these groups.

While there are many possible naming strategies that can be used for groups, it is important to be as consistent as possible particularly with ref groups. While not required, these examples also endeavor to minimize embedded punctuation and use camelCase to simplify the reading of long names, using an underscore (i.e. "_") is another common approach.

The examples below are illustrations of "policy groups and static application permissions," as defined in the Access Control Models section.

Example Access Policy 1 - Computing Lab

Policy 1.0 - All students can log on to the computers in the lab.

An access policy group, app:computerLab:labLoginAuthorized is created in Grouper that maps to the lab computer access control mechanism. The ref:student:allStudents group is then added to app:computerLab:labLoginAllow. There is nothing in app:computerLab:labLoginDeny for this example. app:computerLab:labLoginAuthorized is composite group of labLoginAllow minus labLoginDeny.

Access Policy Group	Allow Groups
app:computerLab:labLoginAuthorized	ref:student:allStudents

Policy 1.1 - All students, all teaching faculty and all computer lab administrators can logon to computers in the lab.

In this example, app:computerLab:labLoginAuthorized has some additional members in the app:computerLab:labLoginAllow group. An institutional reference group for teaching faculty, ref:faculty:teachingFaculty is maintained automatically by institutional data. Additionally, an application scoped reference group for lab administrators, app:computing:computingLabManagers, is maintained manually by the computing lab director. Thus three groups are combined by Grouper into the labLoginAuthorized policy group.

Access Policy Group	Allow Groups
	ref:students:allStudents
app:computerLab:labLoginAuthorized	app:computing:computerLabManagers
	ref:faculty:teachingFaculty

Policy 1.2 - Any member of the university community that has been shown to be violating University Computing Policy shall be blocked from access to University Computing Labs. The first offense shall block usage for two weeks. The second offense shall block usage for the remainder of the current semester. The third offense will permanently block the offender's access until an appeal is approved by the Office of the Chief Information Security Officer.

Implementation of this policy requires the addition of a deny group app:computerLab:labLogin_deny. Any subject with membership in the labLogin_deny group will be denied access regardless of membership in other groups. This policy requires three deny groups to represent the three cases:

- app:computerLab:ref:aupViolationFirst
- app:computerLab:ref:aupViolationSecond
- app:computerLab:ref:aup:ViolationThird

Each deny group is added to labLogin_deny. Membership in any one of the three deny groups would effectively deny access. Membership added to aupViolationFirst would be set to expire within two week either manually or by a Grouper Rule. aupViolationSecond has a Grouper rule that clears all membership at the end of the current semester. aupViolationThird is manually managed by the CISO.

Access Policy Group	Allow Groups	Deny Groups
	computingLabManagers	aupViolationSecond
labLogin_authorized	ref:student:all_students	aupViolationFirst
	ref:faculty:teachingFaculty	aupViolationThird

Example Access Policy 2 - Access to Online Course Material

Policy 2.0 - All students of Introductory Physics will use an online text book and excerpts of classical books such as Newton's Principia via the Physics Department's websites. All students except for physics majors will use the current version of the online book (*physics_101_current*), but physics majors will use the newest version (*physics_101_new*) which is not yet thoroughly reviewed.

This policy calls out **two reference groups:**

- physics majors - ref:student:majors:physics:students
- students enrolled in Introductory Physics - ref:course:*term*:physics:101:students

These two reference groups would be automatically kept in sync with institutional data via the Grouper loader.

The policy also calls out three resources that have different access rules, and these are represented by three **access policy groups**.

- The first one, access to Classical Books would be represented by access policy group, app:physics_books:classicalBooks. The allow group, a pp:physics_books:classicalBooksAllow would have both majors:physics:students and physics:101:students as direct members.
- The second policy states the access to the current version of the online textbook is limited to non physics majors enrolled in the course and is represented by access policy group app:physics_books:physics_101_current. The allow group physics_101_current_allow, would contain the class roster, physics:101:students as a direct member, and the deny group, physics_101_current_deny, would contain physics majors, majors :physics:students.
- The third policy states physics majors taking the course should have access to the newest version of the book, and is represented by access policy group, app:physics_books:physics_101_new. The policy can be implemented with a local app specific reference group that takes the intersection of the class roster and physics majors to create app:physics_books:ref:101_physics_majors. The allow group, physics_101_new_ allow, would have app:physics_books:ref:101_physics_majors as a direct member.

Access Policy Group	Allow Groups	Deny Groups
classicalBooks	majors:physics:students physics:101:students	
physics_101_current	physics:101:students	majors:physics:students
physics_101_new	101_physics_majors	

Previous: Conclusion

Ask the community

It is important to know where you can obtain help or provide feedback before you begin. Grouper is community driven product and the public mailing lists are full of very responsive and experienced users. If you are unsure if you have encountered a bug or are just having issues, you can send an email to the Grouper-users email list. Feedback on the Grouper, such as bugs, suggestions, or feature requests can be put straight into the Grouper JIRA. Whether making a JIRA issue or emailing the list, please include as much information as possible such as version and patches you are running, any relevant logs, any relevant configuration, what you were doing at the time, and what you are trying to accomplish.

- Grouper-Users Email List
  - You can view the archives and subscribe to it here: https://lists.internet2.edu/sympa/arc/grouper-users
- Grouper JIRA Instance
  - https://bugs.internet2.edu/jira/projects/GRP/issues/
  - Go to bugs.internet2.edu and you can sign up for an account
- Grouper-Dev Email List
  - This is a good mailing list to be on if you are interested in the actual development of Grouper. Archives and subscribe here:
  - https://lists.internet2.edu/sympa/arc/grouper-dev
- Grouper Slack
  - To be added to the incommon-grouper Slack channel, submit your request at: https://incommon.org/help/