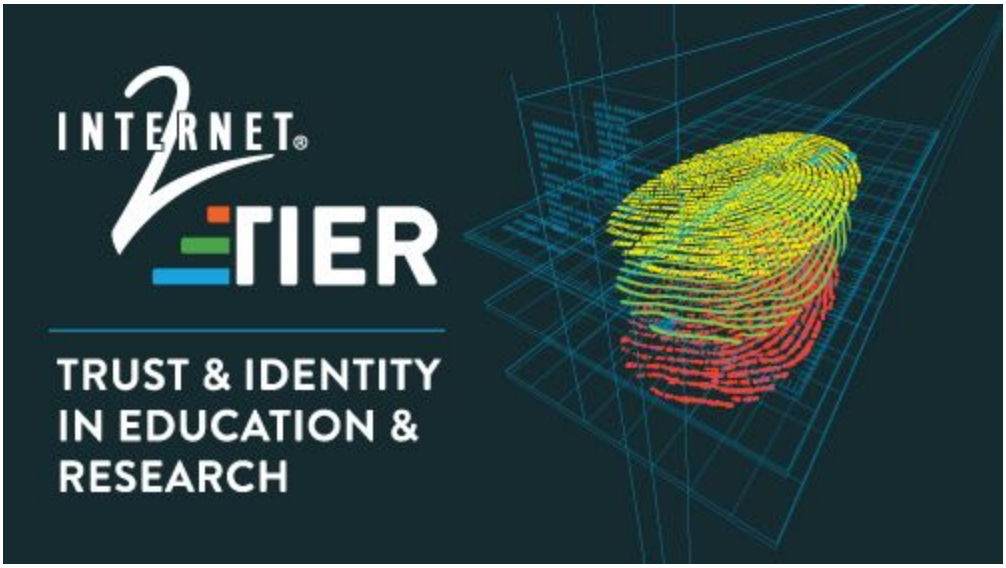
TIER Grouper Deployment Guide

Version 1.0 2017-04-21



Repository ID: TI.25.1
Authors: James Babb
Tom Dopirak
Bill Thompson, Editor
TIER API and Entity Registry WG
Grouper Development Team
Sponsor: Internet2
Superseded documents: (none)
Proposed future review date: April 2018
Subject tags: Grouper, access management, authorization, access control, access control model, access control policy

Table of Contents

List of Figures

1 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 by and large continue to be a mix of institutional 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 a common toolkit for group management and access control governance that is well suited to the needs of the TIER community. The 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 TIER 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 IAM goals more quickly, and work together to build a robust TIER based IAM practice.

The goal of this document 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.

2 Introduction

A TIER based enterprise 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 (e.g. institutional role, access control list, service eligibility, distribution list) management activities.

Grouper can be employed in a variety of flexible access control models, but the underlying approach follows a consistent path. 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.). Institutional cohorts are then turned into reference groups which are used in the digital access policy definition. 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.

NIST 800-162 defines attribute-based access control (ABAC) as, “an access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with subject, object, requested operations, and in some cases environment conditions against policy.” Access policy driven by institutional meaningful cohorts and kept up to date automatically provides one of the primary benefits of an attribute-based access control system. This is described in NIST 800-162 as “accommodating the external (unanticipated user)”, meaning the system can handle users it has never seen before since access policy is based on subject attributes instead of static access control lists or subject to role mappings.

2.1 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

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 using Grouper primarily for achieving access management governance leveraging a subset of Grouper primitives and features. Additionally, Grouper has the ability to manage fine-grained application level permissions, and while those are mostly out of scope for this guide, the access management governance approach described here could be leveraged when incorporating those features.

2.2 Audience

This guide is targeted at both technical staff and managers responsible for identity and access management who are seeking detailed deployment and operational information on a TIER compatible Grouper deployment.

Readers should be familiar with identity and access management concepts and terminology as defined by NIST 800-162 ABAC, 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.

2.3 Document Structure

This rest of the deployment guide is organized as follows:

- Section 3 Understanding Grouper provides a basic understanding of Grouper and how it relates to TIER.
- Section 4 Installing Grouper provides a high-level overview of the major steps required to install Grouper.
- Section 5 TIER Folder and Group Design defines a variety of group types and purposes, and a recommended initial folder and group organization.
- Section 6 Access Control Models describes how TIER and Grouper components come together to achieve access management capabilities.
- Section 7 Provisioning Models discuss provisioning models and strategies.
- Section 8 Operational Considerations provides pointers and tips on operating Grouper in production.
- Section 9 concludes the document.
- Appendix A provides example access policies.
- Appendix B provides acknowledgments.

The guide relies on the Grouper wiki to provide current technical details and version specific information.

2.4 Notes on Terminology

This document builds on and makes use of the NIST 800-162 ABAC terminology as the base for IAM level concepts and borrows Grouper product specific terminology from the Grouper glossary and Grouper 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 TIER/IAM level concepts.

Tip: Make sure you read NIST 800-162 at least three times before moving forward. Really, do it!

Grouper primitives are specific names for features within the product itself. These may be mapped to one or more TIER/IAM level concepts that are used to implement various IAM 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 groups can also be thought of as labels or tags that are applied to subjects. In this way, they can also be viewed as subject attributes from an ABAC perspective.

Basis groups consists solely of direct subject membership assignments and are often maintained automatically by the Grouper Loader process. 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 can be loaded separately into basis groups and then combined into an “employee” reference group.

An **access policy group** is a composite group whose membership is composed of a single include group (i.e. an allow group) and a single exclude group (i.e. 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.

An **account policy group** is a composite group whose membership is composed of a single include group (i.e. an allow group) and a single exclude group (i.e. deny group). Effective membership in an account policy group represents a precomputed account policy decision. Membership within an account policy group signals that a suitable identity record (i.e. an account) should be created and kept in sync with a target service.

Allow/deny groups are used specifically for access policy group definition. Subject membership within an allow group must be indirect via a reference group. That is, the only direct membership assignment permitted in an allow or deny group is a reference group. Reference groups can be institutional in scope or specific to a particular application.

3 Understanding Grouper

A TIER based enterprise 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 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 a TIER based IAM system.

The Grouper project maintains three introductory videos that predate TIER, but are 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:

1. Leverage institutional data to create meaningful cohorts (staff, student, etc)
2. Enable distributed management of exceptions and ad-hoc groups (Institutional Review Board Membership, account locked by CISO (Chief Information Security Officer))
3. Use composite groups to define access policy (allow - deny)
4. Reflect access control decisions to target systems (app:vpn:vpn_authorized)

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 800-162 calls the "natural language policy" (**NLP**). Figure 1 shows how the NLP is translated into digital policy (**DP**) in Grouper.

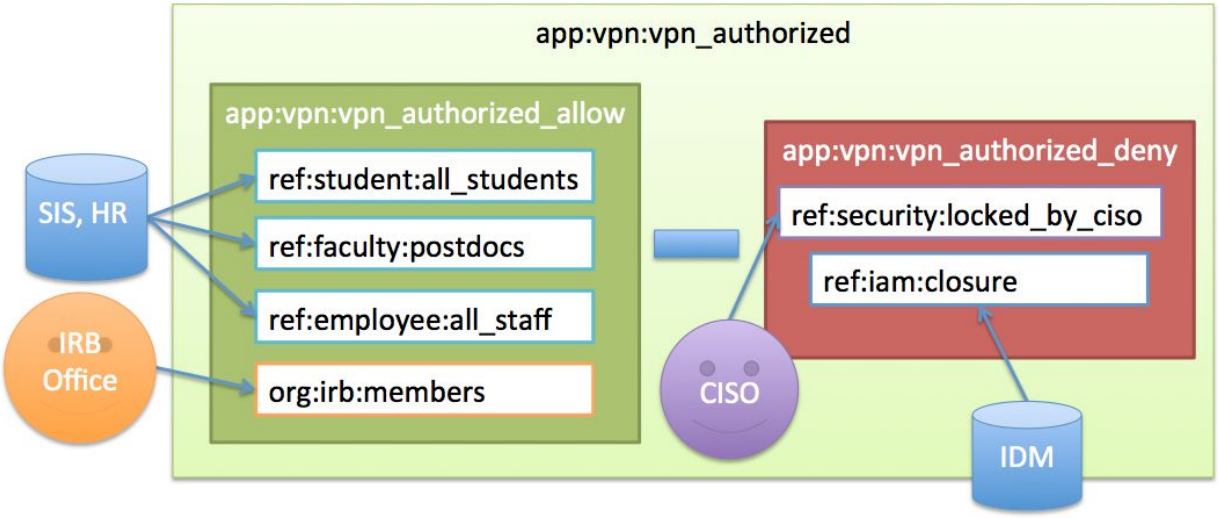


Figure 1: University of Chicago VPN Access Policy

The policy calls out number of different cohorts which we call reference groups. These are groups of subjects that share some characteristic, 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.

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 `vpn_authorized` yields the appropriate digital policy and 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 TIER based Grouper deployment, which is that access policy should be easily discoverable and verified.

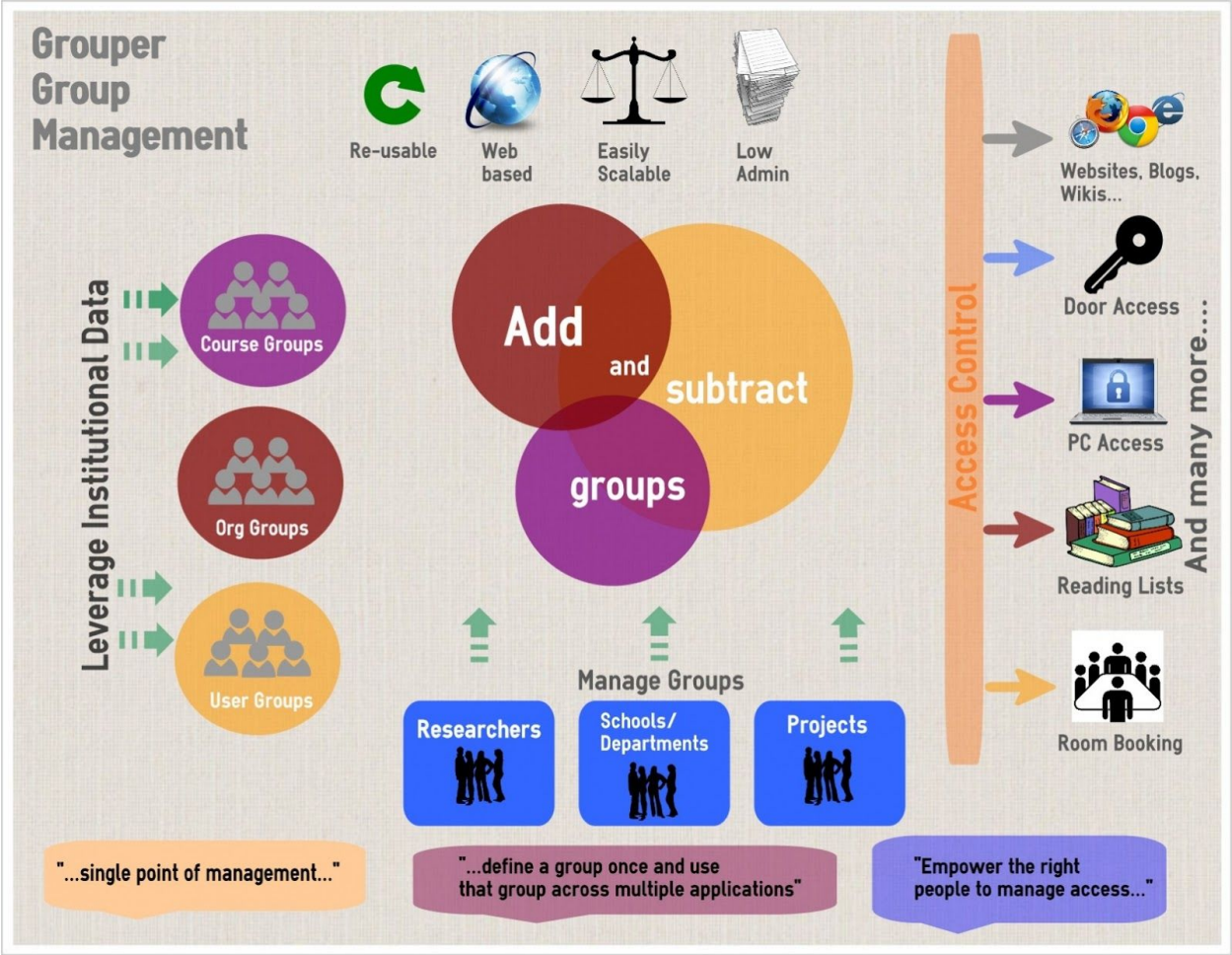


Figure 2: Enterprise Access Management with Grouper[1]

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

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

3.1 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 the 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.

[1] Newcastle University May 2013 Grouper InfoGraphic

Membership in group can be direct or indirect and describes a relationship between a subject or group and a 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.

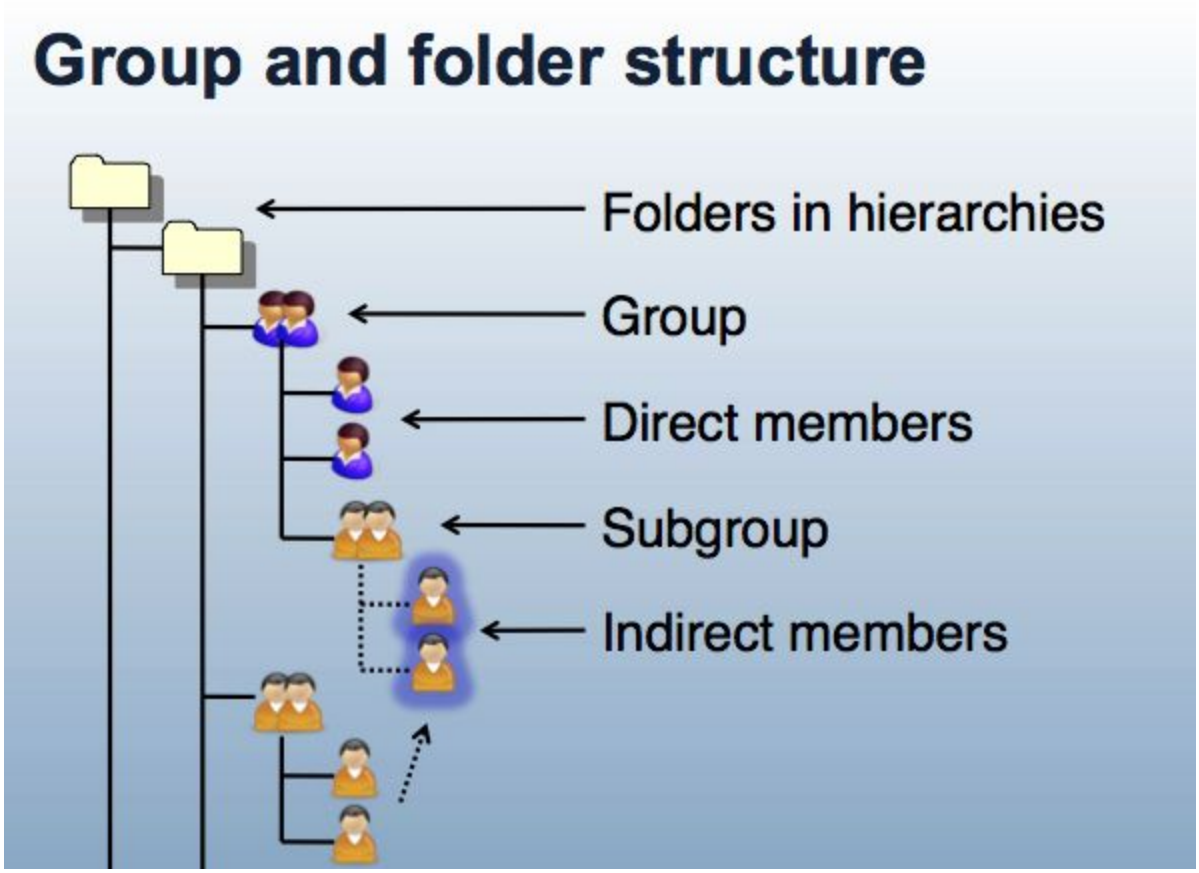


Figure 3: Group and Folder Structure

3.2 Composite Groups

Grouper allows you to use two existing groups to define a third group. The third group called a composite group of the other two factor groups. Groups can be combined as an intersection or complement. 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"). Complement is the primary method used to create composite groups for access policy.

An 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.

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

3.3 Folder and Group Privileges

Folders and groups have privileges that can be assigned to subjects or groups within Grouper. The privilege assignments control who can take what action on a folder or group. Each folder and group has its own privilege assignments which enables fine-grained control and delegation of authority. The Access Privileges definition in the Grouper glossary provides further details on what each privilege provides.

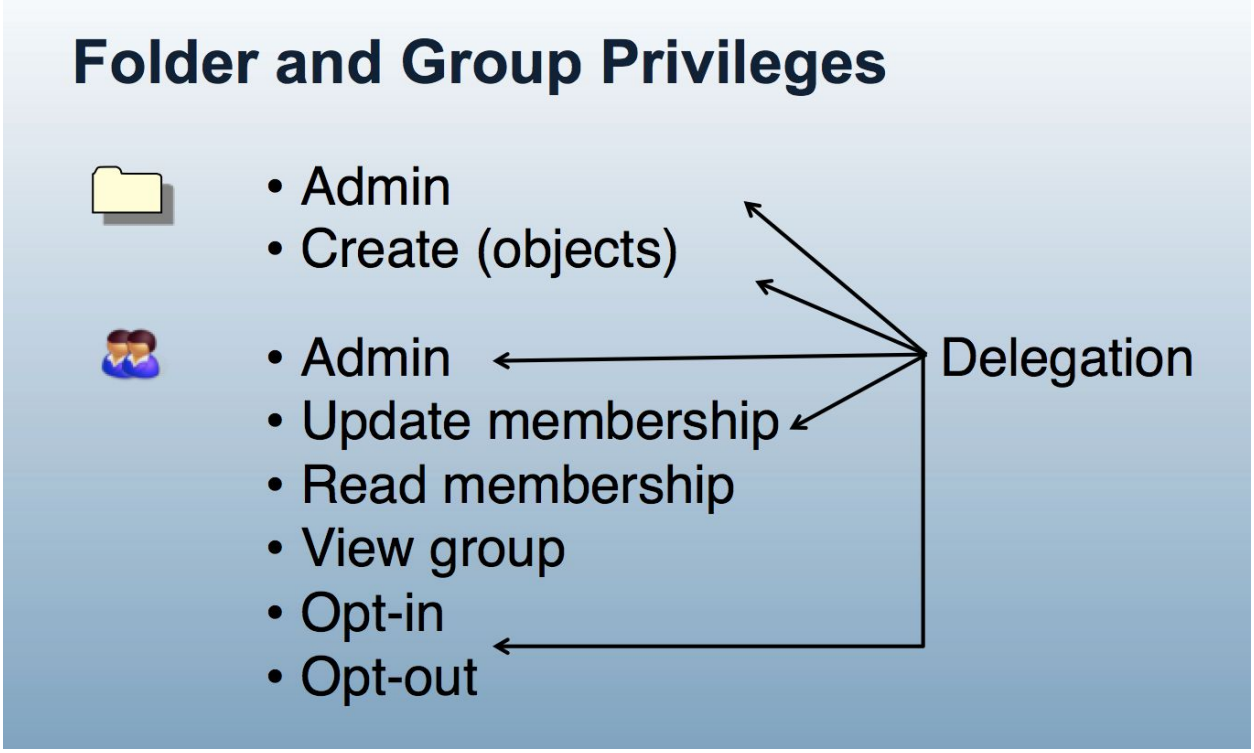


Figure 4: Grouper Privileges and Delegation

The combination of folder hierarchy, administrative groups, and Grouper Rules are used to manage folder and group privileges. Groups and folders created within a parent folder can be configured to 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.

3.4 Grouper Daemon and Loader Jobs

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. Out of the box 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.

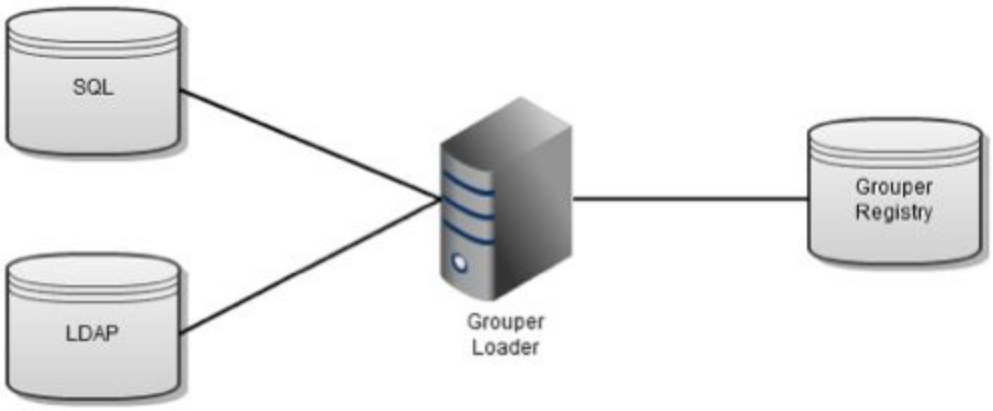


Figure 5: Grouper Loader Jobs

4 Installing Grouper

4.1 Before You Begin

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

4.2 Grouper Installer

Grouper is installed, patched, and upgraded using the Grouper installer. The installer will download and configure all necessary components and will prompt for database connection strings, usernames and passwords, and several configuration elements and deployment choices.

The installer can install all the current Grouper components which includes:

- grouper.apiBinary - used for running the command line client and the Grouper daemon
- grouper.ui - the Java Servlet web-based user interface
- grouper.ws - SOAP and RESTful web service endpoints - used by the Grouper client and for custom integration
- grouper.clientBinary - Java based client library for Grouper LDAP and web services
- grouper.pspng - provisioning plug-in for synchronizing group membership with external systems
- grouper.psp - deprecated provisioning plug-in for synchronizing group membership with external systems

The latest Grouper Installer is available on the Grouper downloads wiki page. The installer requires full schema admin privileges at least for the initial install and upgrades. The installer and gsh (the Grouper command line client) can output DDL for these purposes if necessary to comply with organizational deployment policies. In any case, access to the underlying database is controlled by the Grouper application layer when interacting with grouper.ui and grouper.ws.

Download and run the installer:

- Make sure you have Java 1.7 SDK. Must be the SDK and not just the JRE.
- Make a folder where you want Grouper installed
- Run: `$ java -jar grouperInstaller.jar`

Alternatively, you can pre-stage the install by downloading the required files first and then running the grouperInstaller.jar in the same directory:

- `wget -r http://software.internet2.edu/grouper/release/2.3.0/`
  - move all the files to the target host
- `cd software.internet2.edu/grouper/release/2.3.0`
- `mv patches/grouper_v2_3_0* .`
- `java - jar grouperInstaller.jar`

Most the of defaults can be taken, except:

- `Do you want to use the default and included hsqldb database (t|f)?[t]:` **`f`**
  - database connection string, user and password for target data should be used
- `Do you want to add quickstart subjects to DB (t|f)?` **`f`**
- `Do you want to add quickstart data to registry (t|f)?` **`f`**

- `Do you want to install the provisioning service provider (t|f)? [t]: **f**`

Tip: The installer will initially create the grouper-ui and grouper-ws war files. However, to patch or upgrade, the installer must have access to the exploded war files. Consider building all the components together and moving the grouper.ui and grouper.ws directories to other hosts as needed.

Tip: GrouperSystem is the “root” user and has full administrative privileges. The GrouperSystem password should be strong and treated as a privileged account. Additional subjects can be granted admin privilege by assigning them wheel group membership (which we’ll configure in section 4.4). Add additional members to the wheel group within Grouper before configuring external authentication (such as Shibboleth or CAS). This prevents you from being locked out of the Grouper UI.

Recommendation: Grouper passwords should be externalized and encrypted, especially in production deployments. This includes LDAP and database passwords.

Tip: grouper.installer.properties can be created to run non-interactive Grouper installs.

Tip: To drop all tables and recreate a fresh Grouper database run: “`gsh -registry -drop -runscript -noprompt`”

Tip: Grouper 2.3 installer includes Tomcat 6 which is no longer supported. Recommend deploying on a more recent version of Tomcat. Grouper will work with Tomcat 7 and 8.

4.3 Install gsh Wrapper

The default Grouper shell command line client, gsh, lacks modern shell features and integration with scripting languages such as groovy or python. Shell Wrappers for Grouper provides these features and improves usability and opens opportunities for further operational efficiencies.

4.3.1 Install Groovy

- Groovy install instructions: http://groovy-lang.org/download.html#distro
- verify installation: `$ groovysh -version`

4.3.2 Install Shell Wrappers for Grouper

- `git clone https://github.com/wgthom/groovysh4grouper`
- follow install instructions: https://github.com/wgthom/groovysh4grouper/blob/master/groovy/README.groovy.md
- run `$GROUPER_HOME/grouper.apiBinary-X.X.X/bin/gsh.groovy`
- verify installation: `$ groovy:000> findSubject("GrouperSystem")`

    - `===> Subject id: GrouperSystem, sourceId: g:isa, name: GrouperSysAdmin`
- to exit groovysh: `$ groovy:000> :exit`

Tip: gsh.groovy provides tab completion of all Grouper API objects and a number of convenience function that are defined in groovysh.profile.

4.4 Configure the Grouper API

The Grouper API is the core of the Grouper system, and its configuration controls overall execution and behavior, including the command line interface, gsh, and the Grouper daemon. Configuring the Grouper API is done by collection of properties and XML files.

Grouper uses a configuration overlay method to make it easier to deploy and upgrade Grouper environments. This approach preserves distribution properties files and allows for an override properties files for local configuration.

For an initial development tier install, update *$GROUPER_HOME/grouper.apiBinary-2.3.0/conf/grouper.properties* with the following:

```
# differentiate test vs dev vs prod for logging and emailing
grouper.env.name = dev

# the URL which will be used in emails to users.
#e.g. https://server.school.edu/grouper/
grouper.ui.url = {whatever the URL of dev is}

# auto-created wheel group for convenience
configuration.autocreate.system.groups = true

# A wheel group allows you to enable non-GrouperSystem subjects to act
# like a root user when interacting with the registry.
groups.wheel.use = true
```

4.5 Configure the Subject API

Grouper uses a component called the Subject API to search for and resolve various entities (i.e. subjects, groups, etc). Any subject that you want to assign group membership to must be resolvable via the Subject API. This typically means connecting Grouper to either an enterprise directory such as OpenLDAP or to a relational database used for identity management. Configuration of source adapters provides detailed considerations and install instructions. The Penn subject source JDBC2 example provides details on configuring the Subject API for a SQL database.

Tip: Grouper expects the Subject ID to be unchangeable and irrevocable. Usually this an opaque id (e.g. number or uuid). The source that a subject is associated with also should not change. Subject Identifiers on the other hand are anything that can refer to a subject uniquely. Subject Identifiers typically are usernames, NetIDs or EPPNs. It would be nice if subject id's and identifiers are unique across sources, though this is not required. You should not have the same subject in more than one source.

Tip: Subjects should be resolvable for as long as you want users to be able to search for them or view them on the UI. It is possible for subjects to be inactive in which case they are not searchable, but are still resolvable so they can be shown in the UI for auditing.

4.6 Verify Source Adapter Configuration

- run `$GROUPER_HOME/grouper.apiBinary-X.X.X/bin/gsh.groovy`
- `groovy:000> subj = SubjectFinder.findByIdOrIdentifier("{UNI}", false)`
    - replace {UNI} with a resolvable Id or Identifier like:
      `SubjectFinder.findByIdOrIdentifier("wgt123", false)`
- `groovy:000> subj.getAttributes()`
    - should return all attributes configured in sources.xml

Tip: Make sure search terms referenced in Subject API configuration are indexed.

Tip: *`$GROUPER_HOME/grouper.apiBinary-2.3.0/logs/grouper_error.log`* will have detailed information in the event the configuration is not working.

Tip: Grouper installer configures Tomcat to run the expanded war files for grouper.ui at *`$GROUPER_HOME/grouper.ui-X.X.X/dist/`* and the grouper.ws at *`$GROUPER_HOME/grouper.ws-X.X.X/grouper-ws/build/dist/grouper-ws`*. The classpaths are *`$GROUPER_HOME/grouper.ui-X.X.X/dist/grouper/WEB-INF/classes`* and *`$GROUPER_HOME/grouper.ws-2.2.2/grouper-ws/build/dist/grouper-ws/WEB-INF/classes`*. Here you will find the respective copies of sources.xml and other configuration files.

Tip: Grouper logs for the grouper.ui and grouper.ws defaults to *`$TOMCAT_HOME/logs/grouperUi`* and *`$TOMCAT_HOME/logs/grouperWs`*

Once you verify the sources.xml configuration with gsh.groovy, the configuration needs to be copied to the classpath of the grouper.ui and the grouper.ws webapps. You can also re-run the installer to accomplish the same task. After a restart of Tomcat, you should be able to log in to the grouper.ui with the GrouperSystem account and search for subjects in the IDM database.

Tip: A new administrative screen for Grouper admins to troubleshoot and verify subject API sources has been added and is on by default in a 2.3.0 patch or 2.3.1+. Details on using this can be found in Grouper subject API diagnostics in UI.

4.7 Grouper UI Authentication

The Grouper UI can delegate to Shibboleth or CAS for authentication. Grouper Training - Admin - UI - Part ½ and Newcastle University - Protecting UI With Shib provide details on configuring the Shibboleth SP with Grouper.

4.8 Grouper WS Authentication and Authorization

The grouper.ws by default uses container based authentication and expects users to have a role assignment of “grouper_user”. Users of the web service must be resolvable subjects by the Grouper Subject API. Other options for authentication are possible, and are listed in Grouper WS authentication.

Generally, the steps to configure authentication and authorization are:

1. Create a service account (non-person DN and password) in LDAP
2. Configure container managed authentication in Tomcat that will authenticate the user and return the “grouper_user” role

Tomcat 8 JNDIRealm Configuration for a user with an attribute memberOf=grouper_user:

```
<Realm className="org.apache.catalina.realm.JNDIRealm"
       connectionURL="ldaps://localhost"
       userBase="ou=people,dc=example,dc=edu"
       userSearch="(uid={0})"
       userSubtree="true"
       connectionName="cn=admin,dc=example,dc=edu"
       connectionPassword="password"
       userRoleName="memberOf" />
```

Tip: For development purposes the tomcat-user.xml can also be used to set up a local user/password and roles. However, the subject must still be resolvable by Grouper via the Subject API.

4.9 Grouper Daemon

The Grouper daemon is a command line process which handles many tasks including running Grouper loader jobs, executing change log consumers, validating Grouper Rules, and updating enabled/disabled flags. The Grouper Daemon must be configured properly and always be running in functional Grouper setup. For Grouper Daemon configuration options, see the daemon section of the Grouper API.

Tip: Additional Grouper daemon tasks and options, such as change log consumers and daily reports are also configured in grouper-loader.properties. See grouper-loader.base.properties for the full set.

Tip: If deploying on Linux the Grouper daemon can be setup to run as a linux service so that it auto-starts when the machine boots up.

Tip: To check if Grouper loader daemon is running: "`ps -ef | grep gsh | grep loader`"

4.10 Grouper Loader

The Grouper loader is the primary way group memberships are kept up to date with institutional data. Loader jobs can be driven by SQL or LDAP queries. The Grouper Loader is configured in grouper-loader.base.properties and grouper-loader.properties. SQL and LDAP sources for loader jobs can be configured here, such as a connection to an IDM database or an operational data store of user attributes.

Tip: To add a new SQL data source loader jobs, in grouper-loader.properties add:

```
#################################
## DB connections
#################################
# specify the db connection with user, pass, url, and driver class
# the string after "db." is the name of the connection, and it should not have
# spaces or other special chars in it. pass should be encrypted and
externalized
db.idm.user =
db.idm.pass =
db.idm.url =

# Loader jobs can omit the subject source id if a default is specified
default.subject.source.id = idm
```

SQL Loader jobs come in two flavors; **SIMPLE_SQL** and **SQL_GROUP_LIST**. **SIMPLE_SQL** jobs maintain membership for a single group. **SQL_GROUP_LIST** can create an entire hierarchy of folders and groups and is typically used to maintain things like class rosters and organizational hierarchies.

Tip: Grouper Loader logs the results of loader jobs in the Grouper database in the `grouper_loader_log` table. Grouper loader also logs to *`grouper_error.log`*. *`grouper_error.log`* logging levels is controlled by *`log4j.properties`*.

Tip: The group audit report shows how and when membership changes over time.

4.10.1 SIMPLE_SQL Loader Jobs

SIMPLE_SQL loader jobs provide an easy way to pull in a single reference or basis group from an external database. SIMPLE_SQL Loader jobs are configured directly on groups which are created “in-line” within the Grouper folder hierarchy (e.g. ref:student:classes:class_2020)

Once the loader job sources are available, you can configure a new loader job with the admin UI. Using the web UI, the basic steps are:

1. If needed, create a folder to hold the reference group (e.g. “`ref:student:classes:`”)
2. Create a new reference group (e.g. “`ref:student:classes:class_2020`”)
3. Edit the group to mark it as a grouperLoader job
4. Edit Attributes on the group to configure the loader job
    a. `grouperLoaderDbName = idm`
    b. `grouperLoaderQuartzCron = 0 5 7 * * ?` (7:05am every day)
    c. `grouperLoaderQuery = select subject_id from student_v where class = “2020”`
    d. `grouperLoaderScheduleType = CRON`
    e. `grouperLoaderType = SQL_SIMPLE`

To test the new loader job using the gsh.groovy, run the job:

1. start the gsh.groovy
2. `job = GroupFinder.findByName(grouperSession, "ref:student:classes:class_2020");`
3. `loaderRunOneJob(job);`

This should return something like: “`loader ran successfully, inserted 288 memberships, deleted 0 memberships, total membership count: 288`”

Recommendation: Create SQL views for membership assignments so that Grouper loader job configuration is simple and stable. This way you can change the SQL query in the view without having to update Grouper configuration.

Recommendation: Do not use include/exclude for reference groups. If possible rely on the loader source systems for “truth” about membership in core institutional cohorts.

Tip: As of Grouper 2.3, if you create a new loader job, you can have it scheduled without restarting the daemon by using an option in the New UI. If you are an admin of the group, under "More actions", there is an option named "Schedule loader process". This option can also be used if you change the job schedule.

4.10.2 SQL_GROUP_LIST Loader Jobs

SQL_GROUP_LIST loader jobs provide the capability to create multiple groups and folders from a single query. The result set of the SQL query must contain a group_name in addition to

subject_id. Unlike SQL_SIMPLE, SQL_GROUP_LIST loader jobs are configured in separate “loader groups” rather than the target groups themselves.

SQL_GROUP_LIST are often used to create organizational hierarchies and class rosters. The Grouper loader clas list example from Penn provides a robust example of combining SQL_GROUP_LIST loader jobs and include/exclude groups to provide up to date class membership for students, instructors, assistants (to instructors), guests, and all members. Guests are ad-hoc groups, and provide an entry point for subjects that is not dependent on institutional source systems (i.e. SIS system).

Recommendation: Keep loader jobs that create multiple folder/groups in “`etc:loader:`”.

Tip: The GrouperLoaderGroupsLike configuration options tells Grouper to remove managed groups that no longer have membership assignments returned by the SQL query.

Tip: If your loader jobs are taking too long, consider incremental updates using Grouper loader real time updates. This feature allows you to have incremental loading of memberships for SQL based loader jobs.

The following example SQL query can be used to inspect the GROUPER_LOADER_LOG.

```
--
-- SHOW summary for changelog TEMP rate
--
select
  to_char ( sum (TOTAL_COUNT) / (sum (MILLIS)/1000), '9999.99') as AVERAGE_PER_SECOND,
  to_char ( sum (TOTAL_COUNT), '9999999') as TOTAL_CHANGES,
  to_char ( count ( MILLIS ), '999999') as TOTAL_JOBS,
  to_char ( sum (MILLIS) / 1000, '999999.99' ) as TOTAL_SECONDS
  from grouper_loader_log
  where ( job_name like '%CHANGE_LOG_ch%')
             -- or job_name like '%_consumer_pspng%' )
    and ENDED_TIME is not null
    and (STARTED_TIME >= trunc(sysdate-1) -- today and yesterday
  )
  and TOTAL_COUNT > 0
;
--
-- SHOW entries for changelog TEMP and rate
--
select to_char(STARTED_TIME, 'yyyy/mm/dd-HH24:MI:SS') as STARTED_TIME,
        to_char(ENDED_TIME, 'yyyy/mm/dd-HH24:MI:SS') as END_TIME,
        to_char(MILLIS / 1000,'99999.99') as Seconds,
        to_char(TOTAL_COUNT / (MILLIS / 1000),'9999.999') as RATE,
        SUBSTR(JOB_NAME,1,80) AS JOB_NAME, TOTAL_COUNT AS TOT,
        INSERT_COUNT AS INS, UPDATE_COUNT AS UPD, DELETE_COUNT AS DEL,
        UNRESOLVABLE_SUBJECT_COUNT AS UNRES, STATUS,
        HOST, JOB_TYPE, JOB_MESSAGE
```

```
from grouper_loader_log
where
  ( job_name like '%CHANGE_LOG_ch%' )--or job_name like '%_pspng%')
  --and job_name like '%pspng%'
  and (STARTED_TIME >= trunc(sysdate-1)
  --or JOB_MESSAGE is not null
  )
  --and UNRESOLVABLE_SUBJECT_COUNT > 0
  and TOTAL_COUNT > 0
order by STARTED_TIME  DESC
;

select count(*) from grouper_change_log_entry_temp; -- see backlog count
select count(*) from GROUPER_LOADER_LOG; -- see backlog count

--
-- DETAIL log for selected entries
--
select to_char(STARTED_TIME, 'yyyy/mm/dd-HH24:MI:SS') as STARTED_TIME,
to_char(ENDED_TIME, 'HH24:MI:SS') as END_TIME, MILLIS / 1000 as Seconds,
        to_char(TOTAL_COUNT / (MILLIS / 1000),'9999.999') as RATE,
        SUBSTR(JOB_NAME,1,80) AS JOB_NAME, TOTAL_COUNT AS TOT, INSERT_COUNT AS INS,
UPDATE_COUNT AS UPD, DELETE_COUNT AS DEL,
        UNRESOLVABLE_SUBJECT_COUNT AS UNRES, STATUS, HOST, JOB_MESSAGE
  from GROUPER_LOADER_LOG
  where STARTED_TIME >= trunc(sysdate-1) and
    (
    ((JOB_TYPE = 'CHANGE_LOG') and (STATUS != 'SUCCESS'))
        --(STATUS != 'SUCCESS' OR TOTAL_COUNT > 0) )
         --and job_name = 'CHANGE_LOG_changeLogTempToChangeLog'
    OR (JOB_TYPE != 'CHANGE_LOG'
    --and job_message is not null
    --and lower(job_name) like '%maintenance%s'
    and job_name not like 'subjobFor%'
    OR (job_name like '%_pspng_%' and TOTAL_COUNT > 0)
    --and job_name like '%:Service-%'
    --and STATUS != 'SUCCESS'
    --and (TOTAL_COUNT != 0 )-- or UPDATE_COUNT !=0 or DELETE_COUNT != 0)
    --and UNRESOLVABLE_SUBJECT_COUNT > 0
    ) )
  order by ENDED_TIME DESC, STARTED_TIME DESC
;
```

4.10.3 LDAP Loader Jobs

LDAP loaders jobs can be very handy, especially if you already have institutional cohorts maintained in LDAP or subject attributes available that could be used to drive reference group

membership. A typical Grouper deployment might source some or all its reference groups initially from LDAP, and transition those to sources systems over time.

LDAP Loader jobs come in three flavors; **LDAP_SIMPLE**, **LDAP_GROUP_LIST**, and **LDAP_GROUPS_FROM_ATTRIBUTES**. **LDAP_SIMPLE** jobs maintain membership for a single group. **LDAP_GROUP_LIST** and **LDAP_GROUPS_FROM_ATTRIBUTES** can create an entire hierarchy of folders and groups. Grouper Training - Admin - Loader - Part 2/2 provides further details on setting LDAP Loader jobs.

5 TIER Folder and Group Design

"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

Once Grouper is deployed it is up to the IAM analyst to construct and organize the appropriate folders and groups necessary to achieve the desired access management capabilities. The TIER 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 an organized service growth and promotes effective reuse of common reference groups.

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

5.1 Group Definitions

5.1.1 Basis Groups

Often the best source of data for building institutional meaningful cohorts is a combination of arcane employee/payroll/student codes, often sourced from multiple source 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 institutional meaningful cohorts that are required for access policy. Access policy does not reference 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.

5.1.2 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 institutional 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. 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.

Recommendation: 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 jobs.

Tip: 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 (i.e. what services do "students" have access to?).

5.1.3 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.

Limiting policy groups to indirect membership assignments via reference groups ensures that as subject attributes change, effective membership is 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 within 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.

5.1.4 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. an allow group) and a single exclude group (i.e. deny group). Effective membership of an account policy group represents a precomputed account policy decision.

5.2 TIER Standard Folder Set and Pattern

The TIER recommended standard folder set and pattern consists of seven top level folders that are generally organized by group type, intended use and visibility.

Top level folders:

- 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”
- bundle: - sets of reference groups used in policy for many services
- app: - enterprise applications access control policy - specific policy for a service
- org: - delegated authority, ad-hoc groups, org “owned” apps or reference groups
- test: - test folder for system verification

5.2.1 etc:

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.

5.2.2 basis:

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

5.2.3 ref:

The ref: folder is strictly 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:non-employee: - types of non-employees eligible for services
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

5.2.4 bundle:

Bundle groups are a type of reference group designed to make it easier to manage access to a set of services that are always provisioned for a set of related cohorts. The idea is that you bundle a set of reference groups that are commonly used in access policy across of number of services. Rather than adding the individual cohorts to each access policy, you simply add the bundle group. This allows you to control access policy across many services with a single reference group.

bundle:default_services[_group] - faculty, staff, students, community members, exceptions
bundle:employee_services_group - faculty, staff, consultants/contractors, exceptions

Using “show group’s membership in other groups” you can see what services are being authorized for a bundle group.

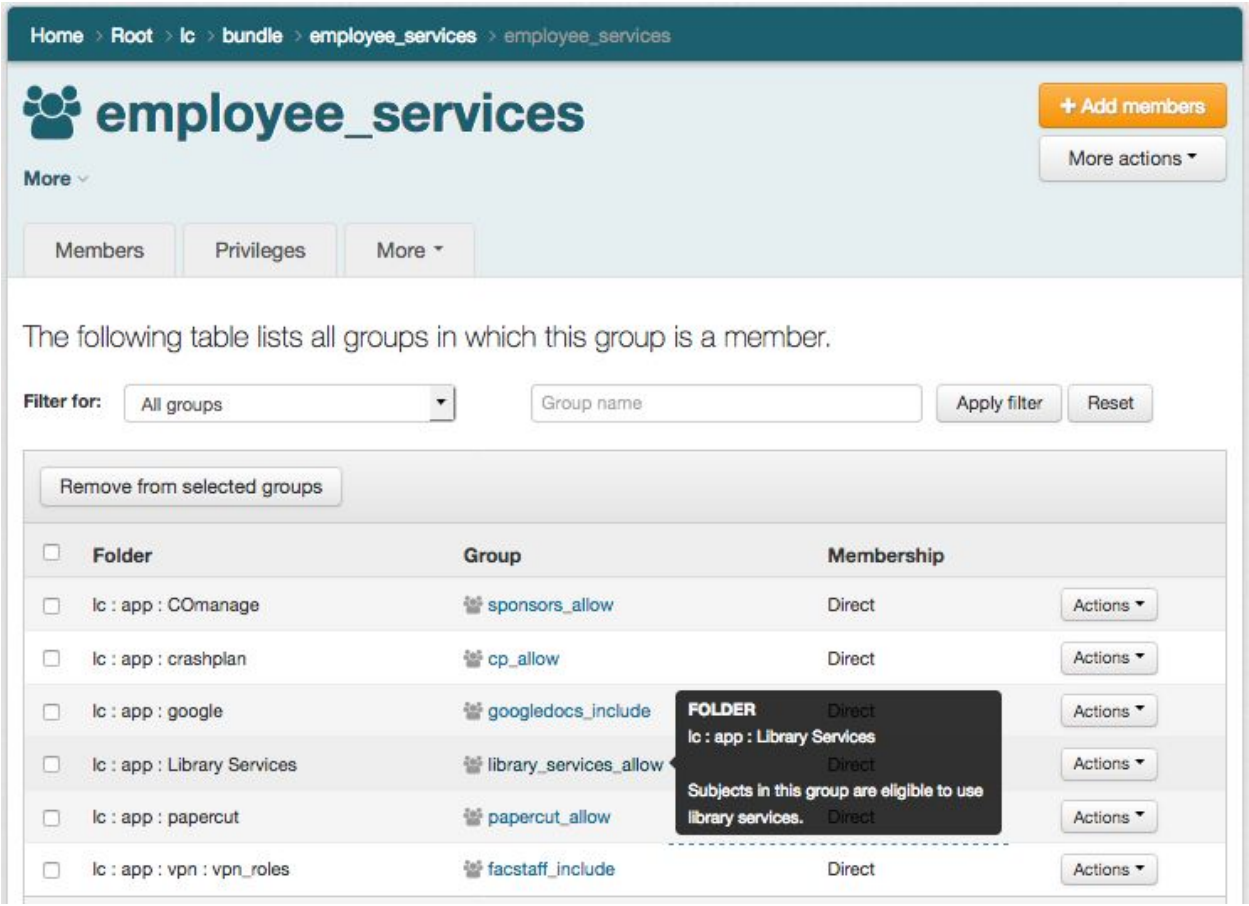


Figure 6: Bundle Group in Access Policy

5.2.5 app:

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. An application named “foo” would have the folder structure:

app:foo: - root folder for the “foo” application
app:foo:etc: - folder for administrative security groups for this application folder and group set
app:foo:etc:foo_admin - members have root-like privileges for the app:foo: folder and group set
app:foo:ref: - folder for “foo” application specific reference group if needed
app:foo:foo_user - access policy group, composite group (foo_users_allow - foo_users_deny)
app:foo:foo_user_allow - only direct members are reference groups
app:foo:foo_user_deny - may include ref:iam:global_deny

5.2.6 org:

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

5.2.7 test:

The test: folder is strictly for Grouper system verification testing and is used solely by the IAM analysts.

6 Access Control Models

The overall approach to access management with Grouper is to create and maintain institutional meaningful cohorts (reference groups), which in turn are used to drive access policy groups. The access policy groups then provide subject attributes (role names or entitlements) that are mapped to coarse or fine-grained permission sets at the target service.

How the fine-grained application permission sets are managed is usually specific and local to the target service. In some cases, the privilege to use a particular service (a set of rights to specific resources) can be mapped to a subject attribute representing an entitlement (i.e. subject is entitled/authorized to access the service). In these cases, a membership assignment in an access policy group can drive an eduPersonEntitlement value that is often consumed by the target service via a SAML assertion. In other cases, group membership must be provisioned to the target service to effectively control access.

How application permission sets are managed, membership assignments are communicated, and access policy is enforced can vary quite considerably depending on the security needs and capabilities of the target service. However, the overall approach to access management with Grouper remains consistent. The following sections use terminology and models from ABAC 800-162 and XACML to demonstrate a variety of models leveraging this approach.

6.1 Access Control Model 1 - Grouper Subject Attributes

In this model, Grouper is used to master subject attributes that represent some type of affiliation or status at the institution. Actual access policy is completely local to the service, and the availability of an agreed upon subject attribute is sufficient to make a policy decision. The

subject attribute policy is a group configured in Grouper and built up with reference groups similar to an access policy group, but with no particular service in mind.

PAP, PDP, and PEP are all at the SP:

1. A subject attribute like eduPersonAffiliation is mastered in Grouper and reflected into an LDAP based enterprise directory.
2. The subject attribute is either passed to the target service via SAML or looked up via LDAP after authentication.
3. PAP: Access control policy is configured at the target service
4. PDP, PEP: Policy decision point and the policy enforcement point are all done at the target service

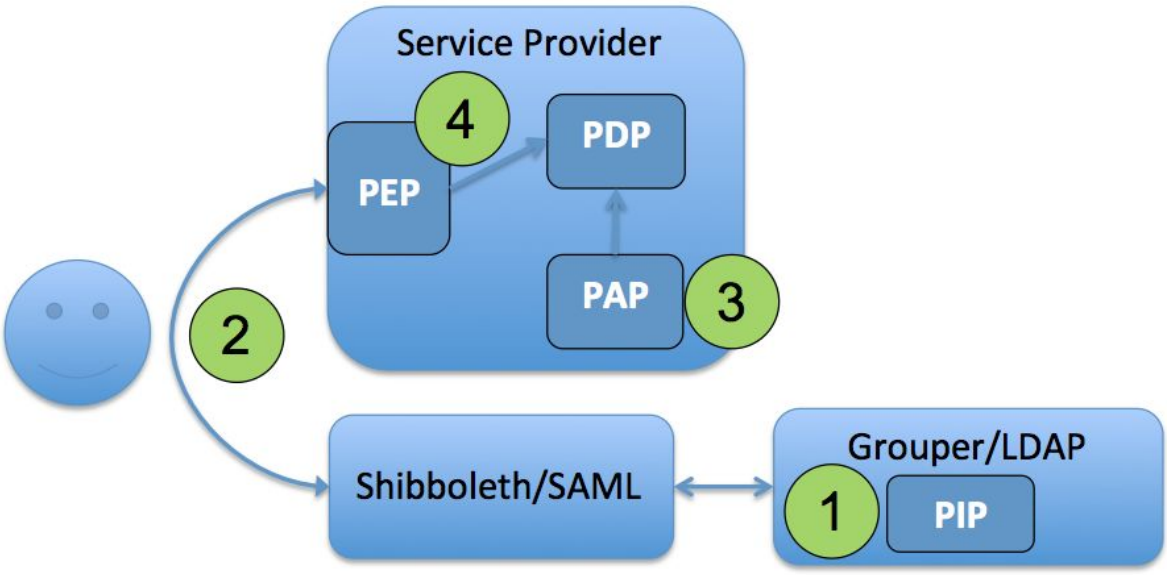


Figure 7: Access Control Model 1 - Grouper Subject Attributes

This model is useful for cases where there exists a loose relationship between the institution and the service provider. Assuming both are in a federation like InCommon, and a locally defined notion of eduPersonAffiliation is sufficient for access control, a broad set of services can be enabled fairly easily.

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 cohorts, instead consider ACM2 and service specific access policy.

6.2 Access Control Model 2 - Grouper as PAP and PDP

In this model access policy administration and the policy decision point (precomputed membership assignments) are done in Grouper and can be communicated to the target service in a variety of ways. Access policy groups in Grouper enables direct mapping from natural language policy to digital policy and back, and policy is kept up to date automatically as subject attributes change. This model supports audit, compliance, and attestation. It applies to both coarse-grained access control and limited/static application roles.

PAP, PDP done in Grouper - PEP done at SP

1. PAP: Access policy groups configured in Grouper based on institutional meaningful cohorts (i.e. reference groups)
2. PDP: Policy decisions (membership assignment in access policy group) precomputed and reflected into LDAP based enterprise directory as LDAP groups or eduPersonEntitlement values.
3. PEP: Policy decision communicated to the PEP at the target service via SAML, LDAP query, direct Grouper WS query, or provisioning to the service.

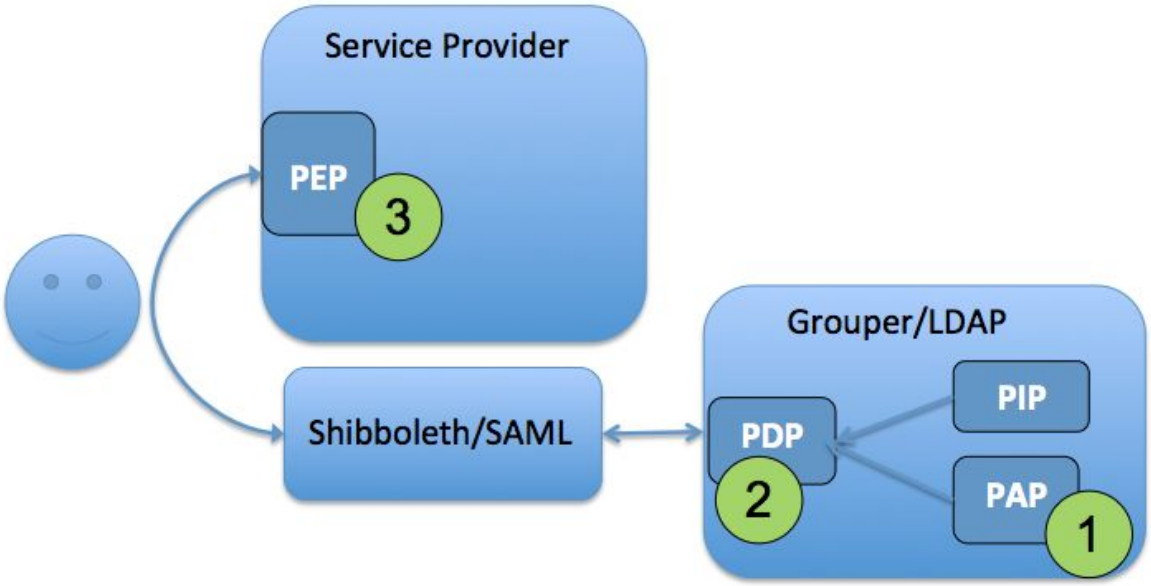


Figure 8: Access Control Model 2 - Grouper as PAP and PDP

ACM2 is applicable across a broad range of services where access control policy can be based on subject attributes, the policy decision can be precomputed, and simple subject attributes are sufficient to drive the enforcement point. The VPN example from the Introduction is an example of the model.

6.3 Access Control Model 3 - RBAC User to Role Mapping

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 name 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 control group with the role name defined at the target service. The policy of which subjects are mapped to application roles (permissions sets) can be attribute based or a simple access control list, or some combination of both.

Subject -> Role assignment in Grouper - Permission -> Role at service

1. Fine-grained permission sets are managed at the target service (Role -> Permissions) and assigned a Role Name
2. Grouper access control group statically mapped to target service Role Name and provides User -> Role mapping
3. PAP split between Grouper and target service, PDP and PEP at service

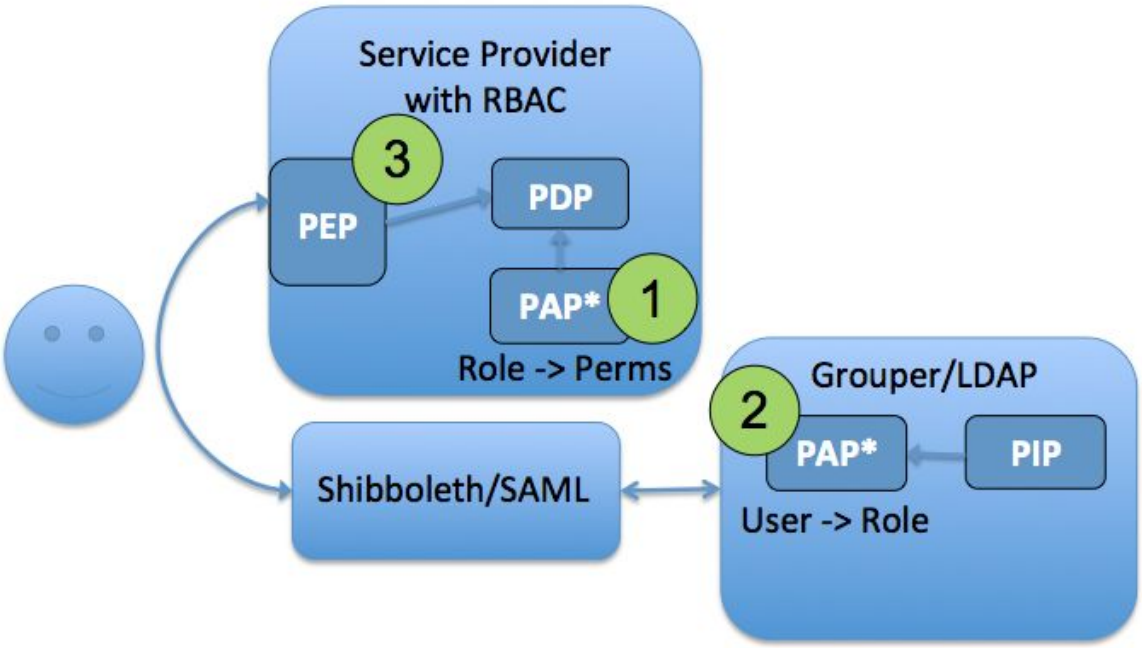


Figure 9: Access Control Model 3 - RBAC User to Role Mapping

For ACM3 to work well the target system needs a mechanism to associate the Grouper group with the role name configured in the applications RBAC system. This could be done with custom integration of the Grouper client or by making the group available in LDAP. The mapping

between Grouper group and application role name is what enables the User -> Role mapping to stay in sync with policy.

6.4 Access Control Model 4 - WebSSO Short-circuit

Often for pragmatic reasons one wants to limit access to a target service by locating the policy enforcement point at the web single sign-on layer. Two 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, and 2) when the target service lacks a good unauthorized user experience. In the first case, inserting a PEP within the authentication flow enables Grouper to overlay policy control. In the second case the user experience can be improved by redirecting the user to a more helpful “unauthorized” page rather than receiving a cryptic error page from the target service.

PAP, PDP done in Grouper, PEP at WebSSO

1. PAP: Access policy groups configured in Grouper based on institutional meaningful cohorts (i.e. reference groups)
2. PDP: Policy decisions (membership assignment in access policy group) precomputed and reflected into LDAP based enterprise directory as LDAP groups or eduPersonEntitlement values.
3. PEP: WebSSO flows honor policy decision by short-circuiting authentication flows to an a generic or application specific “unauthorized user” page.

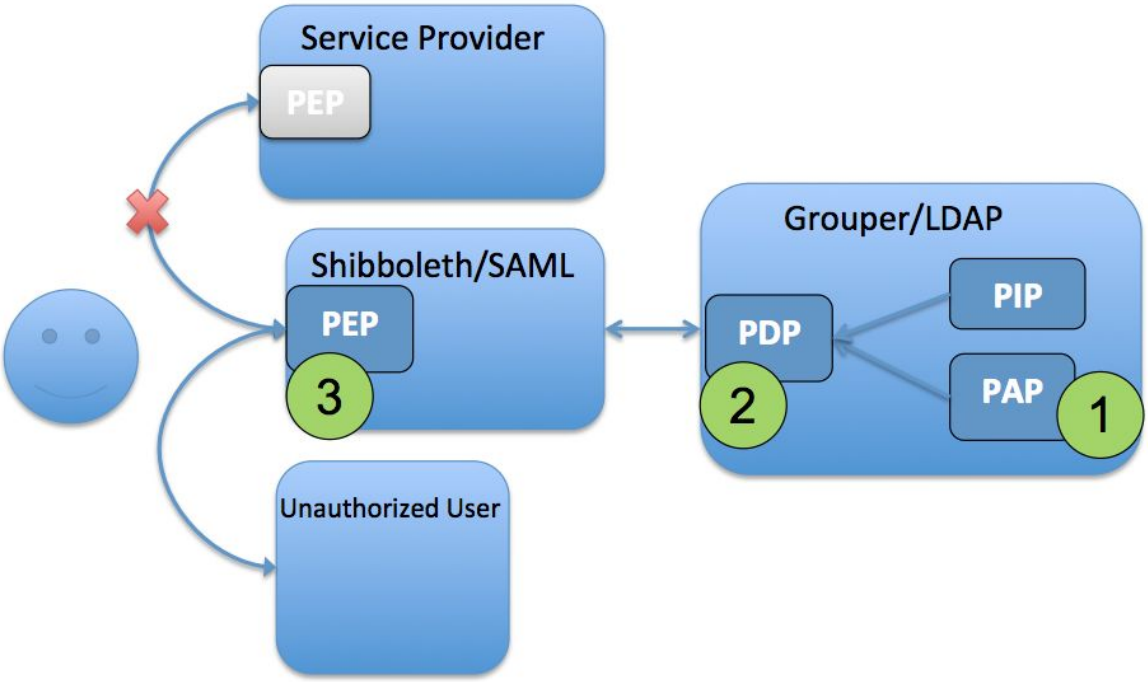


Figure 10: Access Control Model 4 - WebSSO Short-circuit

It should be noted that ACM4 does not absolve the application from also having to do some form of access control.

Recommendation: Do not commingle “unauthorized user” pages with the WebSSO deployment. Keeping them separate will make maintenance and upgrades easier. If possible the “unauthorized user” page should be the responsibility of the application owner.

6.5 Distributed Access Control Management

All the models described above 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 general pattern is to include reference groups whose management has been delegated to the appropriate parties. This can take form of application specific reference groups, organizational reference groups, or identity lifecycle and security groups. Just as with institutional reference groups, membership changes in the delegated groups are immediately reflected in access policy.

6.6 Application Permissions Management - RBAC with Grouper

All of the access control models discussed so far assume the target application has some mechanism to manage application specific permissions. This could be a fairly static application roles to permissions mapping, or a sophisticated RBAC like system one might find in ERP or business intelligence system. In all cases the target system itself is responsible for its own permission management.

While mostly out of scope for this guide, Grouper does have an advanced RBAC system 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. From an application architecture perspective, Grouper becomes a subsystem of the overall application. Grouper fulfills all application permission management needs, and can be integrated with the access control models previously discussed.

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

7 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 setup to keep various reference groups in sync with downstream 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 two categories. The first category, “direct from Grouper to target service”, covers Grouper specific components and plugins for various targets such as AD/LDAP, Duo, etc. The second category, “message queue based delivery”, is a successful strategy employed by many campuses. This strategy relies on a message queue infrastructure to communicate membership and group changes in Grouper to appropriate provisioning components. 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.

8 Operational Considerations

Tools and topics for ongoing administration provides a reference for a variety of operational tasks as does ongoing administration tasks. Grouper Training - Admin - Maintenance - Take 1 video also goes into further details on common operational tasks.

Depending on your needs you may want to consider various deployment architectures highlighted in Grouper high availability.

Grouper diagnostics and Grouper report provide capabilities for monitoring the health of your deployment.

Recommendation: Externalize and encrypt Grouper passwords to improve the security posture of your deployment.

Tip: 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.

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

9 Conclusion

This document builds on industry standards such as NIST 800-162, and provides TIER specific terminology, concepts and strategies required for a TIER compatible Grouper deployment based on a synthesis of current Grouper community practices.

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

Appendix A - Example Access Policies

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 institutionally managed (ref:), maintained by an organization (org:), or application specific reference groups (app:). 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.

Example Access Policy 1 - Computing Lab

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

Solution 1.0 - An access policy group, `app:computerLab:labLogin_authorized` is created in Grouper that maps to the lab computer access control mechanism. The `ref:student:all_students` group is then added to `app:computerLab:labLogin_allow`. There is nothing in `app:computerLab:labLogin_deny` for this example. `app:computerLab:labLogin_authorized` is composite group of `labLogin_allow` minus `labLogin_deny`.

Access Policy Group	Allow Groups
`app:computerLab:labLogin_authorized`	`ref:student:all_students`

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

Solution 1.1 - In this example, `app:computerLab:labLogin_authorized` has some additional members in the `app:computerLab:labLogin_allow 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 `labLogin_authorized` policy group.

Access Policy Group	Allow Groups
	`ref:students:all_students`
`app:computerLab:labLogin_authorized`	`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.

Solution 1.2 - 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.

Solution 2.0 - This policy calls out two reference groups; physics majors and students enrolled in Introductory Physics. The physics majors reference group, `ref:student:majors:physics:students`, and the course roster reference group `ref:course:`*`term`*`:physics:101:students` 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. The first one, access to Classical Books would be represented by access policy group, `app:physics_books:classical_books`. The allow group, `app:physics_books:classical_books_allow` 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 physic 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
classical_books	majors:physics:students physics:101:students	
physics_101_current	physics:101:students	majors:physics:students
physics_101_new	101_physics_majors	

Appendix B - Acknowledgements

This guide is the result of a TIER community collaborative effort, and was made possible by the many contributions from the TIER API and Entity Registry WG, the Grouper Development Team, and many others.

Special thanks to Tom Dopirak for the Example Access Policies, Michael R Gettes for the SQL query to inspect the GROUPER_LOADER_LOG, Newcastle University for the Grouper Infographic, Emily Eisbruch for curating the Grouper community contributions, and Columbia University for the Columbia Grouper deployment guide.

Albert Wu - UCLA Bert Bee-Lindgren - Georgia Tech Bill Kaufman - Internet2 Bill Thompson - Lafayette College Brian Savage - Boston College Brian Woods - Rice Carey Black - The Ohio State University Chris Hyzer - Penn Dean Lane - Rice Emily Eisbruch - Internet2 Eric Goodman - UCOP Ethan Disabb - University of Florida Ethan Kromhout - UNC Chapel Hill Gabor Eszes - Old Dominion Gary Brown- University of Bristol Harry Samuels - Northwestern James Babb - UW Madison Jill Gemmill - Clemson Jim Fox - University of Washington	Jon Finke - RPI Jon Miner - UW Madison José Cedeño - Oregon State University Keith Hazelton - UW Madison Keith Wessel - University of Illinois Ken Koch - Washington University Maarten Kremers - SURFnet Mark McCahill - Duke Michael Gettes - Penn State Michael Hodges - University of Hawaii Mike Zawacki - Internet2 Paul Caskey - Internet2 Raoul Sevier - Harvard Rob Carter - Duke Scott Cantor - The Ohio State University Shilen Patel - Duke Steve Carmody - Brown Steve Moyer - Penn State Steve Zoppi - Internet2 Tom Barton - University of Chicago Tom Dopirak - “Retirement” Tom Jordan - UW Madison Tom Zeller Warren Curry - University of Florida