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.

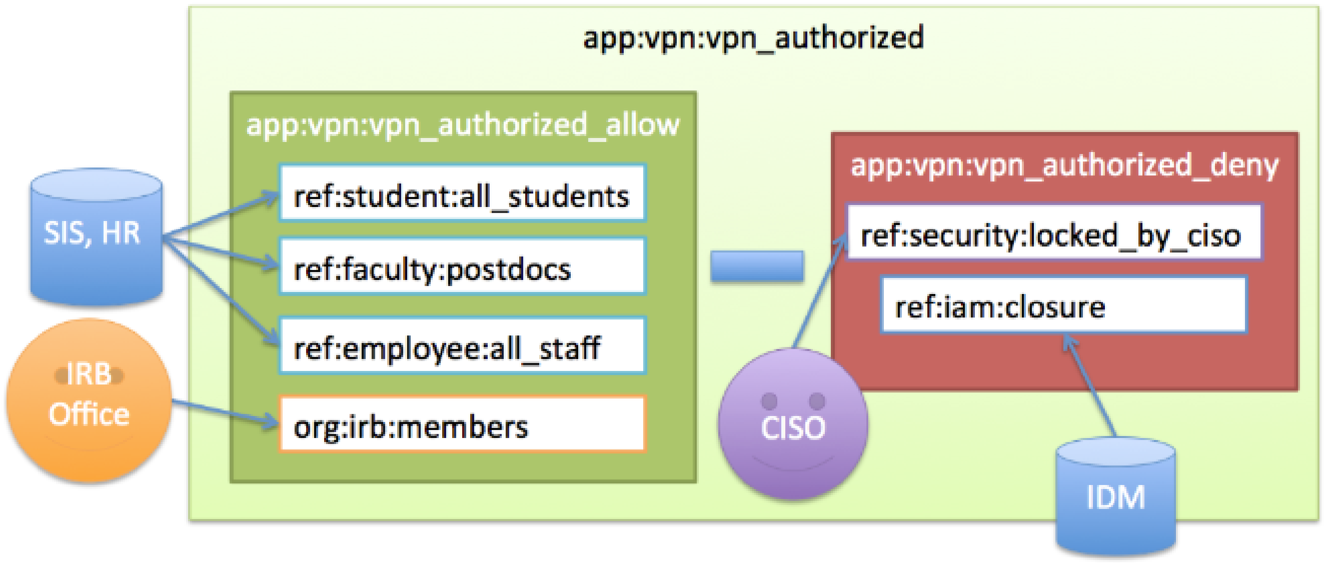


Figure 1: University of Chicago VPN Access Policy

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.

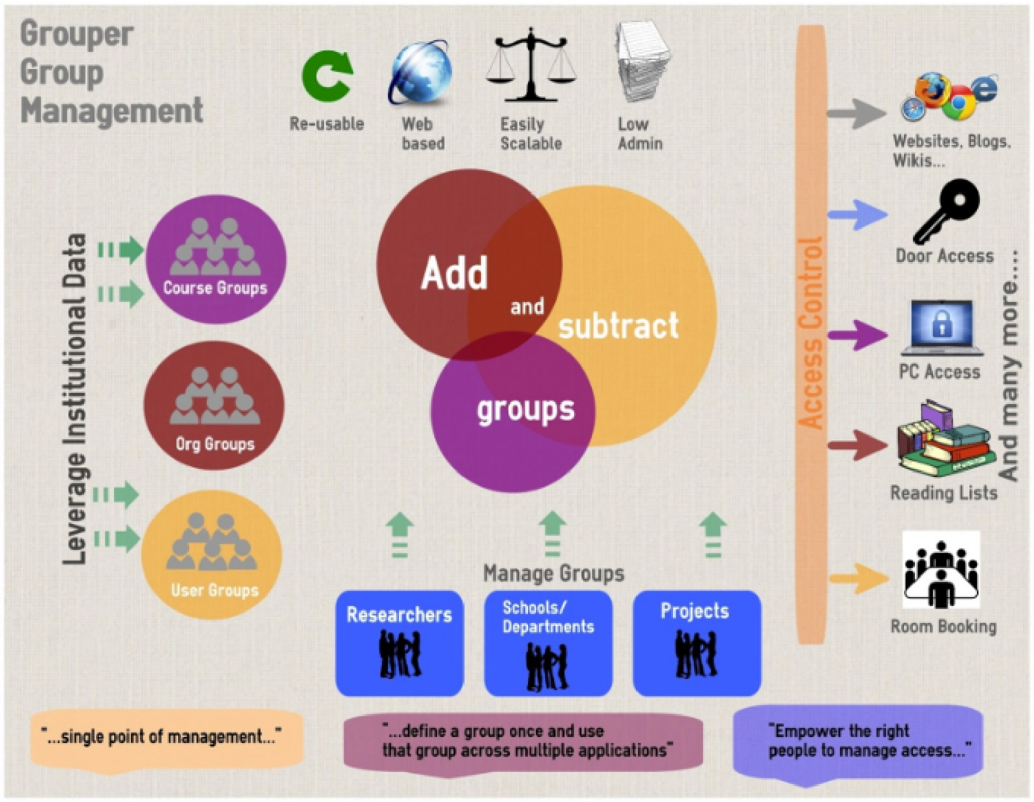


Figure 2: Newcastle University Grouper Infographic

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.

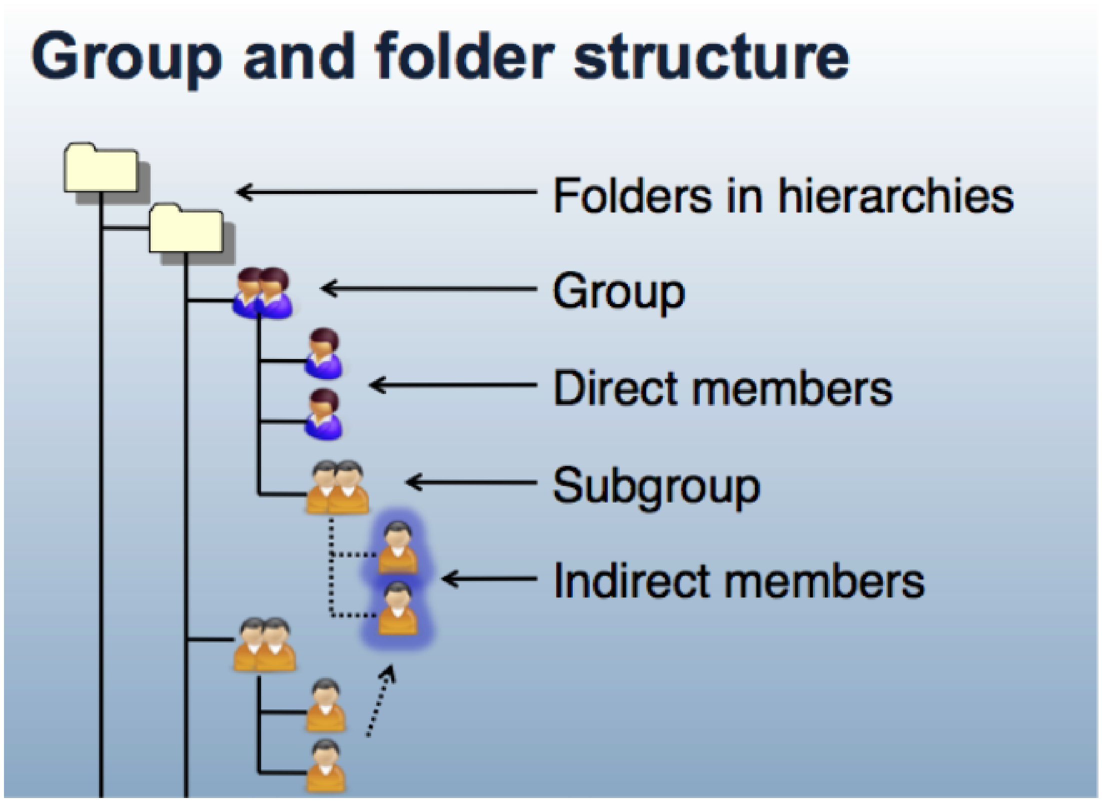
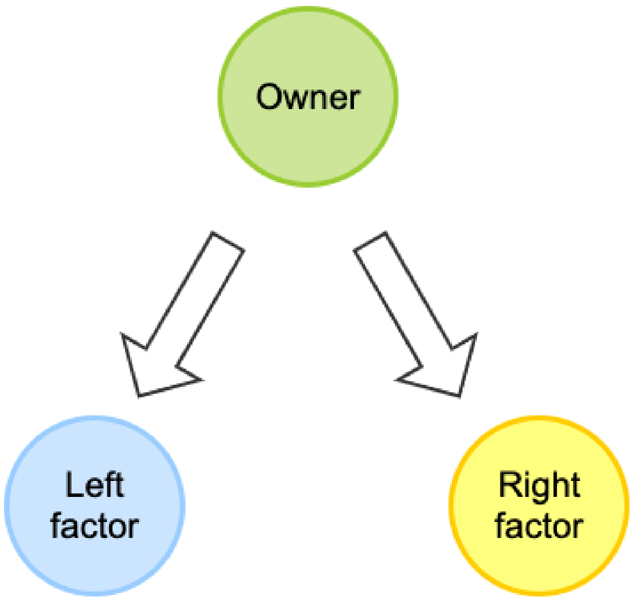


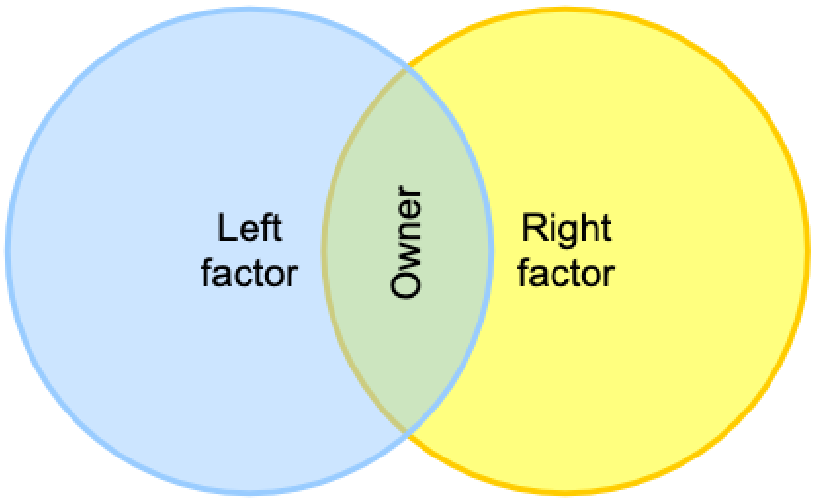
Figure 3: Group and Folder Structure

Composite Groups

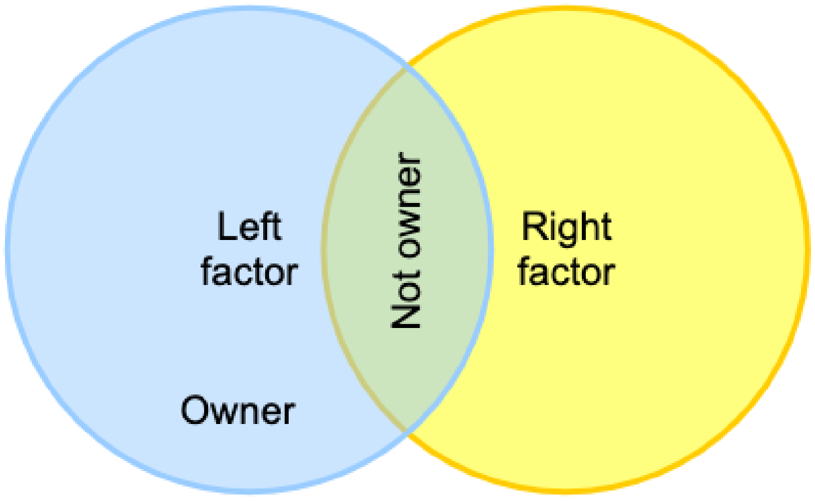
Grouper allows you to use two existing groups to define a third group. The third group is a **composite** of the other two factor groups. Groups can be combined as an **complement** or **intersection**



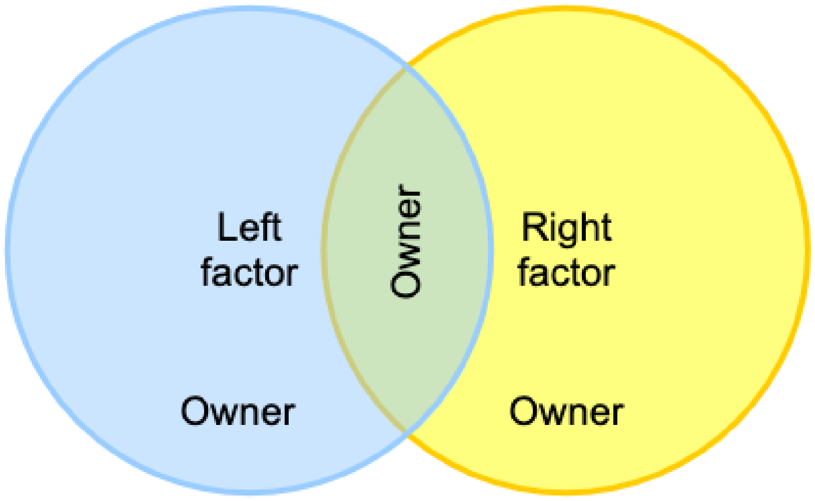
- **Intersection** includes entities that belong to both of the original factor groups, and produces a composite "members-in-common". Intersection groups are often used when creating reference groups from basis groups.



- **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.

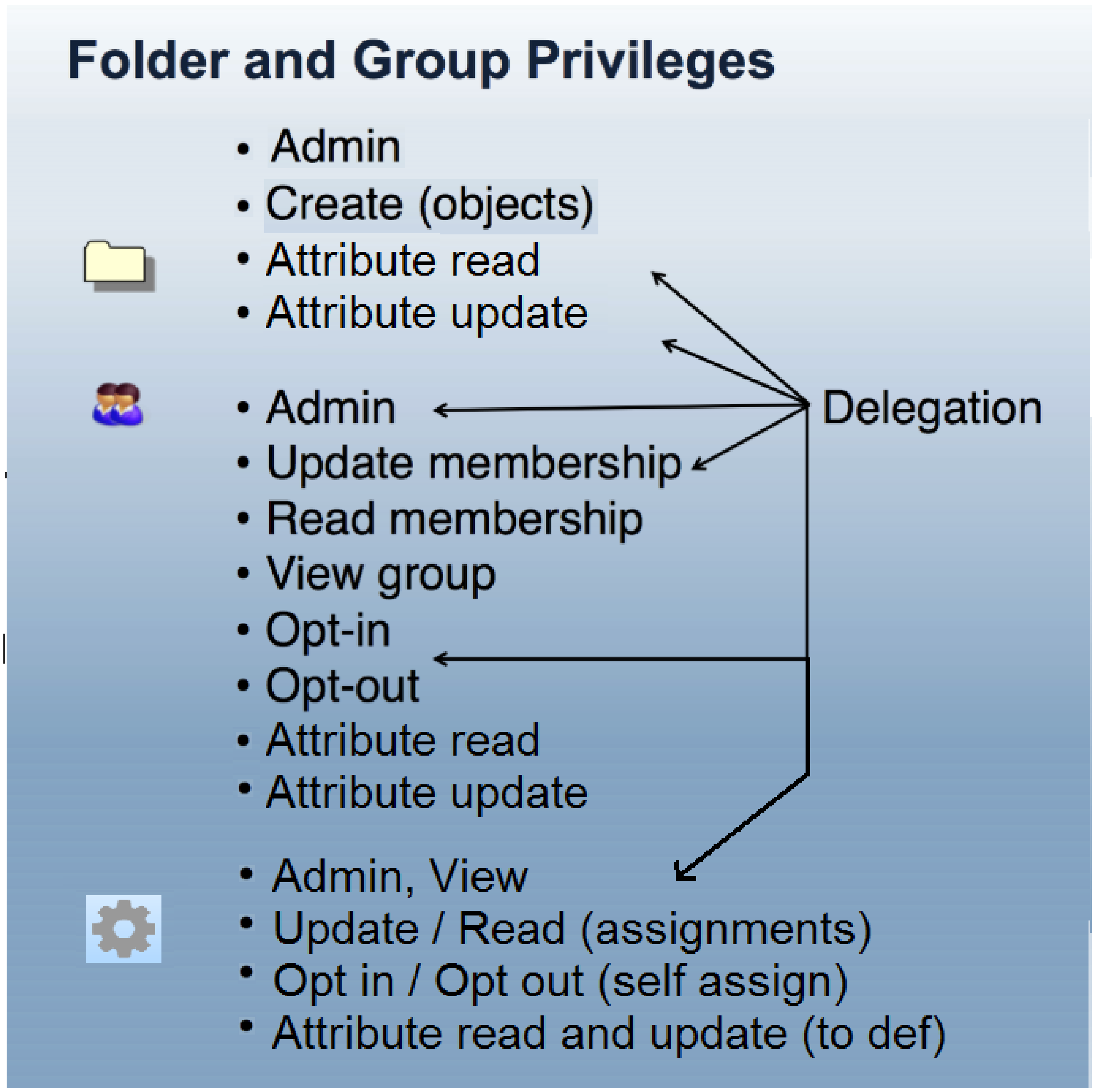


Figure 4: Grouper Privileges and Delegation

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.

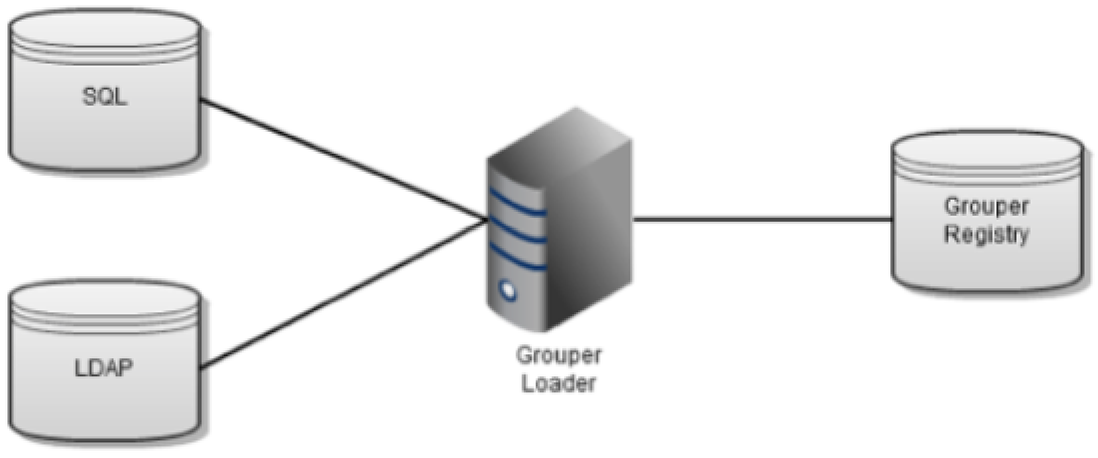


Figure 5: Grouper Loader Jobs

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