Subject API

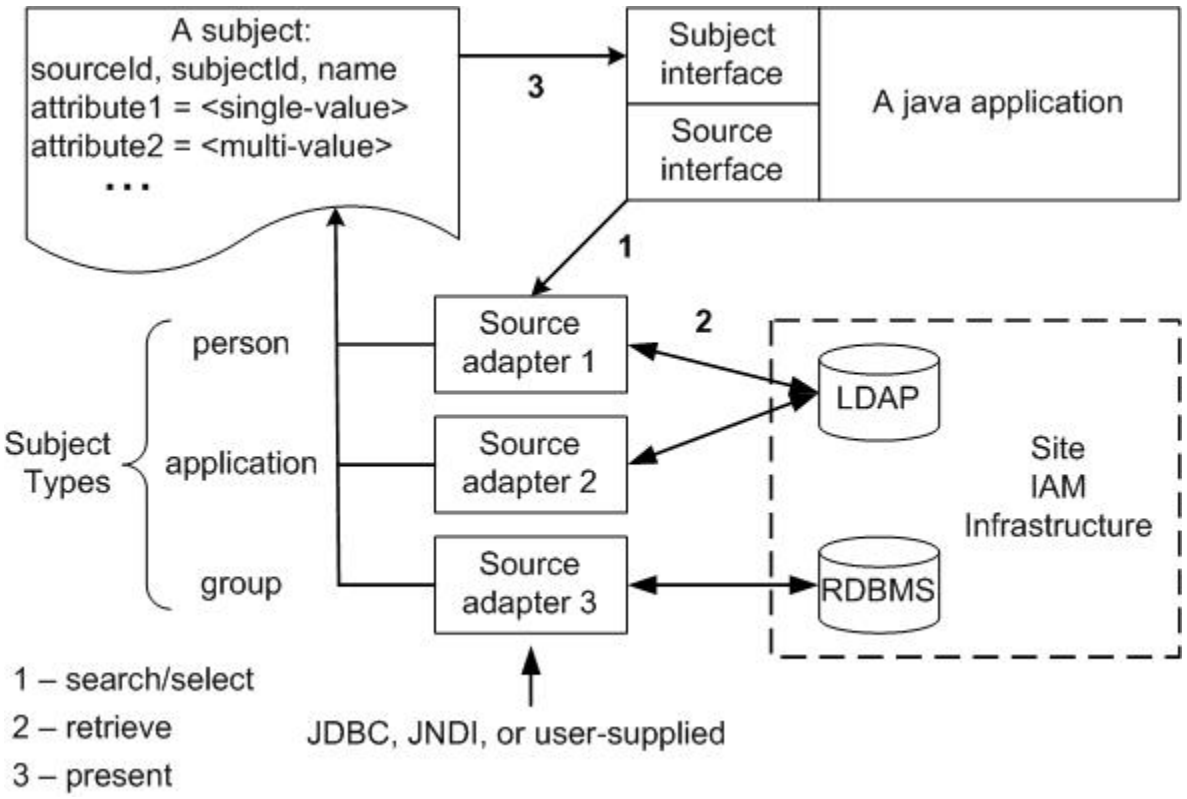
The Subject API

Note: in Grouper 2.3+ you should use the subject.properties not source.xml

The Subject API is used to integrate a java application with a site's existing Identity Management operations (see architectural diagram). It enables any type of object whose identity is being managed - person, group, application, computer, etc. - to be presented to that application without requiring the application to be specifically designed for particular object types or with knowledge of how those objects are stored and represented. Those details form the configuration of the Subject API.

Figure 1 (below) illustrates the general role of the Subject API in the interaction between an application and a site's Identity Management infrastructure. There are two parts to the Subject API: the Source interface and the Subject interface. An application uses the Source interface to search for and select Subjects from back-end stores, which are presented as abstracted, flat Subject objects via the Subject interface.

Figure 1: Subject API Interaction Model



- Grouper local entities
- Grouper subject filter and attribute decorator
- LDAP Subject API example
- Member search and sort columns
- Migrating from the Grouper JDBC subject source to the JDBC2 subject source
- subject-0.2.1-doc
- subject-0.3.1-doc
- subject-1.0-doc
- Subject API search filter by status
- Subject API security by realm
- Subject Identifier column in member table

Debugging

Run the Subject API diagnostics from GSH. Also use the Subject API diagnostics in "misc" in the UI (if Grouper starts... if there is a subject API problem it is severe for Grouper).

```
GrouperSession.startRootSession();
new edu.internet2.middleware.grouper.grouperUi.serviceLogic.SubjectSourceDiagnostics().assignSourceId
("SMUPerson_DEV").assignSubjectId("empl1").assignSubjectIdentifier("netid@school.edu").assignSearchString("em").
subjectSourceDiagnosticsFromGsh()
===>
SUCCESS: Found subject by id in 37ms: 'empl1'
         with SubjectFinder.findByIdAndSource("empl1", "SMUPerson_DEV", false)
SUCCESS: Subject id in returned subject matches the subject id searched for: 'empl1'
WARNING: No subject found by identifier in 14ms: 'netid@school.edu'
         with SubjectFinder.findByIdentifierAndSource("netid@school.edu", "SMUPerson_DEV", false)
```

Note, to debug your SubjectAPI configuration, set this in the log4j.properties.

```
log4j.logger.edu.internet2.middleware.subject.provider = DEBUG
log4j.logger.edu.vt.middleware.ldap = DEBUG
```

If you are using a JDBC source, you can use the p6spy sql driver, set the spy.properties to specify the underlying driver and the log file name (in 2.5 we need to revisit this)

Number of sources

Decide how many sources you need. It should be the minimal number that you can do. For people, it should be one. If you dont have one single source, consider working on that initiative. Having multiple subjects in Grouper that represent the same person will lead to problems (e.g. seeing what someone has access to). You might end up with a source for people and a source for service principals.

Choosing Identifiers for Subjects

Identifiers and their management can get complicated. They can be revoked or not, re-assigned or not, lucent or opaque, etc. Depending on such characteristics, a given identifier might be a good or bad choice to use in the context of managing the identified subject's group memberships.

For example, a username is often lucent - easily remembered by the person to whom it is associated. But it may also be revokable, meaning that it no longer refers to that person (perhaps they have a new one), or even re-assignable, meaning that it might refer to some other person at a later time. If a username is used to record membership, username changes must trigger corresponding membership changes. A username is better suited to authentication than it is to indicating membership.

On the other hand, an opaque registryID (machine, not human, readable) that never changes is great for membership, but lousy for authentication - it might not even be known by the person to whom it is associated. How would I identify myself to Grouper if I wished to opt-in to a list or manage a group?

Grouper accommodates subject identifier issues in two ways. First, it maintains UUIDs for every subject and group within the Groups Registry. These are never exposed by the UI, but are associated with externally supplied subject identifiers within the Groups Registry (in the grouper_members table). This approach allows the identifier associated with a given subject to be changed without any need to change actual memberships.

Second, by relying on the Subject API, Grouper is able to lookup subjects that are presented with an identifier in one namespace and obtain identifiers in other namespaces for that subject. That means that it can translate a username into a registryID, for example. So, when a user authenticates to an application using the Grouper API, that application can use the Subject API to fetch an identifier for the person chosen by the site for use in memberships. Similarly, when a membership in the Groups Registry is to be expressed elsewhere, the identifier used for group members can be translated by a provisioning connector by use of the Subject API into one that is suitable in the provisioned context.

Subject ID: should be unchangeable, unrevokable. Usually this an opaque id (number or uuid etc). The source that a subject is associated with also should not change.

Subject Identifier: anything that can refer to a subject uniquely. Usually these are netIds, eppns, etc.

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.

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 not be active in which case they are not searchable, but still be resolvable so they can be shown in the UI in auditing.

Examples

- Penn JDBC2 example

Search & Selection Methods

The Source interface provides three principal methods of searching for and selecting Subjects. These methods are used in the Grouper API, and are exposed in the UI and WS.

Method	Description
getSubject	Retrieve a specific subject from a specific source by its SubjectId.
getSubjectByIdentifier	Retrieve a specific subject by unique match against one or more configured *identifying attributes*.
search	List all subjects meeting a given search criterion.

Deployers supply back-end specific search & selection statements for each of these three methods that determine 1) when a Subject matches each search criterion and 2) which of its attributes will be presented to the calling application. Callers need only persist a reference to the sourceId and subjectId of Subjects to be able to fully instantiate them at any time. Various methods in the Subject interface provide access to these identifiers and other attributes of each Subject.

The getSubject() method is used by the application to instantiate a Subject object from its persisted subject reference data (subjectId and sourceId). For example, the Grouper UI uses getSubject() to display the name each member of a group.

The getSubjectByIdentifier() method is used to enable the application to locate a unique subject by reference to any of its identifying attributes. For example, consider a site that manages both netIds and registryIds for its users, and suppose they choose to use registryId as their subjectId. When a user logs in with their netId, the application uses getSubjectByIdentifier() to locate and instantiate a Subject object for the user from the user's netId.

The search() method is used by a User Interface application to allow a human to search for and list subjects using familiar attributes like name parts, departments, etc. For example, to grant a person a privilege, the Signet UI first does a search() using the user's specified search term, displays a list of the names and descriptions of the matching subjects, and enables the UI user to select one.

There are attributes that need to be configured for a subject in addition to subjectId:

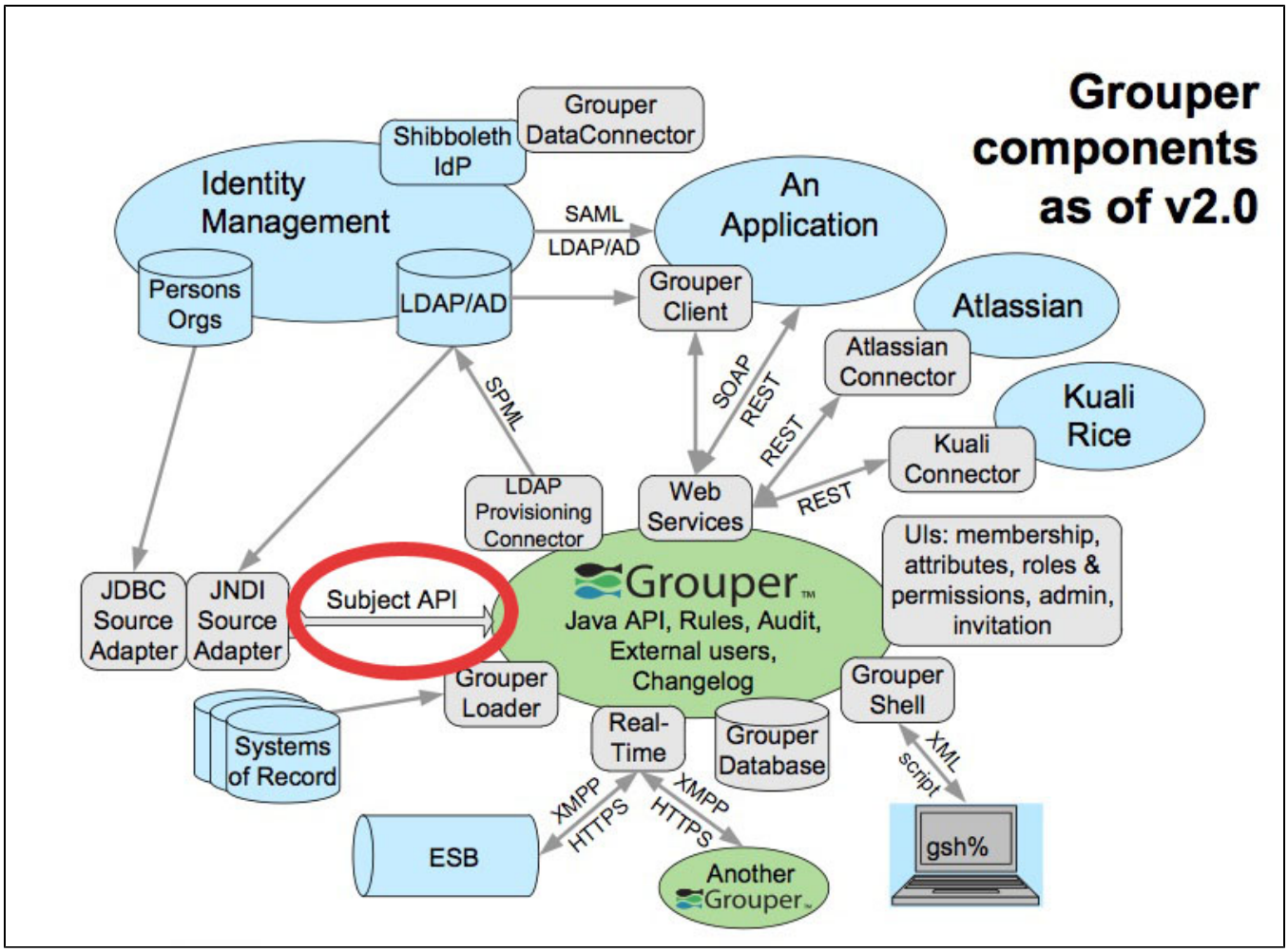
- name: This is generally the first and last name for a subject. If this is private data and you dont want to list it, you can use a netId or something to help differentiate the subject from other subjects. Worst case, subjectId
- description: This should be something that is standalone to show information about the subject when a list is displayed to help the user select the correct subject. This is the description attribute at Penn

```
Chris Hyzer (mchyzer, 10021368) (active) Staff - Isc-applications & Information Services - Application
Architect (also: Alumni)
```

- sdf

The Subject API in Grouper Architecture

Figure 2: Subject API



Documentation

- Subject API v0.3.1
- Subject API v0.2.1
- Subject API v1.0

See Also

Subject API Diagnostics in UI

LDAP Subject API Example