Grouper Provisioning Service Provider (PSP)

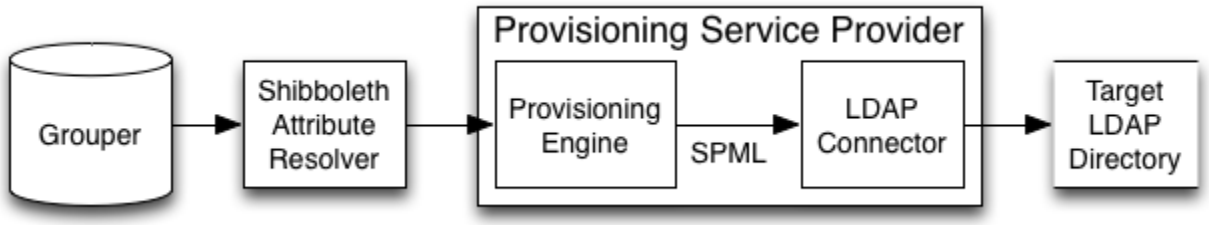
 These topics are discussed in the "Grouper Provisioning Service Provider" training series.

For Provisioning with Grouper 2.3 and above, see this page

See Also

Introduction

Grouper groups, memberships, and stems may be provisioned using the provisioning service provider (PSP, formerly known as LDAPPC-NG).



The PSP serves as an (incomplete) SPMLv2 provisioning service provider, which provisions objects to targets. Objects consist of identifiers, attributes, and references to other objects. Group memberships may be considered to be references. This diagram shows where the PSP fits in the Grouper architecture.

Provisioned objects are calculated from source data returned by the Shibboleth Attribute Resolver. The Shibboleth Attribute Resolver accepts many data connectors (sources), including LDAP, RDBMS, and Grouper.

The psp provisions targets using a standard provisioning language, SPMLv2. An SPMLv2 to ldap connector is provided, based on VT Ldap. Provisioning non-LDAP targets requires a target specific connector, for example, SPMLv2 to RDBMS.

Currently, the psp supports SPMLv2 requests represented as java objects via the Oasis SPMLv2 implementation. The requester is Grouper's cli, gsh.

Usage

The psp may be run using GrouperShell (gsh).

To provision, polling every 60 seconds for changes :

```
bin/gsh.sh -psp -bulkSync -interval 60
```

To calculate how an object should be provisioned :

```
bin/gsh.sh -psp -calc stem:groupName
```

no arguments	Display usage.
`-bulkCalc`	Calculate provisioning for all identifiers.
`-bulkDiff`	Determine provisioning difference for all identifiers.
`-bulkSync`	Synchronize provisioning for all identifiers.
`-calc <id>`	Calculate provisioning for an identifier.
`-diff <id>`	Determine provisioning difference for an identifier.
`-sync <id>`	Synchronize provisioning for an identifier.
`-entityName <id>`	Provisioned object or schema entity id. For example, group, member, etc.
`-interval <seconds>`	Number of seconds between the start of recurring provisioning iterations. If omitted, only one provisioning cycle is performed.
`-conf <dir>`	Path to configuration directory.
`-logSpml`	Log SPML requests and responses.
`-output <file>`	Print SPML responses to output file. Defaults to stdout.
`-printRequests`	Print SPML requests as well as responses.
`-requestID <id>`	SPML request identifier.
`-returnData`	Return data (identifier and attributes).
`-returnEverything`	Return everything (identifier, attributes, and references).
`-returnIdentifier`	Return identifier only.
`-targetID <id>`	Target identifier.

One of `-bulkCalc`, `-bulkDiff`, `-bulkSync`, `-calc <id>`, `-diff <id>`, or `-sync <id>` must be specified. All other arguments are optional.

`calc`

Calculate how an object should be provisioned.

Upon receipt of a `calc` request, the psp will calculate how an object (or objects) should be provisioned, and will return a `calc` response representing the correct provisioning.

`diff`

Determine the changes necessary to transform a provisioned object from how it is currently provisioned to how it should be provisioned.

Upon receipt of a `diff` request, the psp first performs a `calc` request to calculate how objects *should* be provisioned. Then, the PSP queries each target to determine how objects *are* provisioned. The psp returns a `diff` response representing the changes necessary to synchronize the provisioned objects from how it is currently provisioned to how it should be. The changes consist of add, delete, and or modify requests.

`sync`

Synchronize a provisioned object.

Upon receipt of a `sync` request, the psp first performs a `diff` request to determine provisioning changes. Then, the PSP requests targets to perform the changes, and returns the results as a `sync` response.

`bulkCalc|bulkDiff|bulkSync`

Bulk requests operate on all configured source identifiers, and includes the remove of orphan objects (provisioned objects for which no source identifier is known).

LDAPPC-NG -> PSP

The code formerly known as "ldappc-ng" has been refactored to "psp".

module	description
psp	The provisioning service provider and provisioning engine.
psp-distribution	Distribution package.
psp-distribution-for-grouper	Distribution package for Grouper.
psp-example-*	Example configuration and junit test modules.
psp-grouper-changelog	Grouper change log integration including data connectors and change log consumer.
psp-grouper-ldap	Grouper and LDAP integration including attribute definitions to transform Grouper names to LDAP DNs and vice versa.
psp-grouper-source	Data connectors which return the identifiers of all Grouper groups, stems, and members.
psp-grouper-target	Grouper provisioning target.
psp-ldap-target	LDAP provisioning target.
psp-parent	Parent project.

"Real-Time" Provisioning

Real-time provisioning is the provisioning of groups, stems, and memberships triggered from the Grouper change log. In Grouper 2.1 real-time incremental provisioning is available using the PSP. Real-time full provisioning is not yet available. Incremental provisioning is distinguished from full provisioning in that only a single or subset of an attribute's values are provisioned.

Real-time provisioning is available from the provisioning service provider as of Grouper 2.1.0. It should be possible to install and configure the PSP for Grouper versions 1.6 and up.

These instructions assume that Grouper subjects are already provisioned to your LDAP directory, and makes use of the vt-ldap based ldap source adapter.

Known Issues with "Real Time" Provisioning

- The PSP will run the shibboleth attribute resolver for every entry in the changelog including entries for which it will eventually do nothing. This is done because the Shibboleth Attribute Resolver is setup to resolve all attributes prior to returning the data for further evaluation.
- You cannot create an empty group to be incrementally provisioned. The groupOfNames schema that most LDAP and AD setups use **REQUIRES** at least one member. In practice, this means that you must add a member to a group within about 45 seconds of creation of that group or provisioning (and subsequent updates) will fail for that group. The only way to fix a group broken in this manner is to sync that group manually via `gsh -psp -sync group:that:is:broken`. After running that command, the PSP will resume incrementally maintaining the group.

Requirements

- Grouper API (requires Java 6 and a database, details are here)
- provisioning service provider (PSP)
- LDAP directory

Install : Grouper Installer

To install Grouper including the API, UI, WS, grouperClient, psp, etc., download and run the Grouper Installer.

As of Grouper 2.1.0, the PSP may be installed but not configured via the Grouper Installer.

```
curl http://www.internet2.edu/grouper/release/2.1.0/grouperInstaller.jar -O
java -jar grouperInstaller.jar
```

Install : Manual

Install the provisioning service provider by copying jar files and example configuration files from the psp distribution to your Grouper API installation.

Many jars are dependencies of the Shibboleth attribute resolver and may not be necessary in your deployment.

To install manually, download and unpack the psp, then copy jars and configuration files to your Grouper API installation.

Install : Manual - Download and Unpack the PSP

Download the PSP here and unpack.

The distribution name is of the form `grouper.psp-2.1.0.tar.gz`

```
curl http://www.internet2.edu/grouper/release/2.1.0/grouper.psp-2.1.0.tar.gz -O
tar xzf grouper.psp-2.1.0.tar.gz
```

Install : Manual - Copy Jars

Copy jars located in `lib/custom` from the PSP distribution to the Grouper API installation.

```
cp -vR grouper.psp-2.1.0/lib/custom/ grouper.apiBinary-2.1.0/lib/custom/
```

Install : Copy Example Configuration Files

Copy example configuration files located in `conf` from the psp distribution to the Grouper API installation.

Example psp configuration files are in directories named with the prefix `psp-example-*`.

```
cp -vR grouper.psp-2.1.0/conf/ grouper.apiBinary-2.1.0/conf/
```

Configuration Example : Grouper to LDAP

This configuration example should apply to any ldap directory server.

examples	psp-example-grouper-to-ldap
DN structure	bushy
`member`	member DNs

Configuration Example : Grouper to Tivoli

This configuration example targets an IBM Tivoli Directory Server with requirements from Penn State.

examples	psp-example-grouper-to-tivoli
DN structure	flat
`member`	member subject ids
`memberOf`	group DNs
`hasMember`	member names
`isMemberOf`	group names

Configuration Example : Grouper to OpenLDAP

This configuration example applies to OpenLDAP, and includes provisioning the `mailLocalAddress` attribute sourced from the Grouper attribute framework.

examples	psp-example-grouper-to-openldap
DN structure	bushy
`hasMember`	member names
`isMemberOf`	group names
`member`	member DNs
`memberOf`	group DNs

For this example, new attribute framework `etc:attribute:mailLocalAddress` and `etc:attribute:seeAlso` attributes will need to be created in Grouper. Here is example code using the Grouper API :

```
GrouperSession.startRootSession();
Stem etcAttributeStem = StemFinder.findByName(GrouperSession.staticGrouperSession(), "etc:attribute", true);
AttributeDef attributeDef = etcAttributeStem.addChildAttributeDef("mailLocalAddressAttributeDef",
AttributeDefType.attr);
attributeDef.setAssignToGroup(true);
attributeDef.setMultiValued(true);
attributeDef.setValueType(AttributeDefValueType.string);
attributeDef.store();
etcAttributeStem.addChildAttributeDefName(attributeDef, "mailLocalAddress", "mailLocalAddress");

AttributeDef seeAlsoAttributeDef = etcAttributeStem.addChildAttributeDef("seeAlsoAttributeDef",
AttributeDefType.attr);
seeAlsoAttributeDef.setAssignToStem(true);
seeAlsoAttributeDef.setMultiValued(true);
seeAlsoAttributeDef.setValueType(AttributeDefValueType.string);
seeAlsoAttributeDef.store();
etcAttributeStem.addChildAttributeDefName(seeAlsoAttributeDef, "seeAlso", "seeAlso");
```

Configuration Example : Grouper to OpenLDAP MemberOf Overlay

This configuration example applies to an OpenLDAP directory with the memberOf overlay. This example is similar to the OpenLDAP example, except that the memberOf attribute is not provisioned by the psp.

examples	psp-example-grouper-to-openldap-memberof-overlay
DN structure	bushy
`hasMember`	member names
`isMemberOf`	group names
`member`	member DNs

Configuration Example : Grouper to OpenLDAP Multiple

This configuration example provisions multiple OpenLDAP directories, and is based on a request from the University of Modena and Reggio Emilia on the grouper-users@internet2.edu mail list.

https://lists.internet2.edu/sympa/arc/grouper-users/2012-03/msg00005.html

This configuration example provisions two OpenLDAP directories identically while provisioning groups only to a third directory.

Grouper LDAP subjects are sourced from the identical directories allowing for failover.

ldap.properties

```
edu.vt.middleware.ldap.ldapUrl=ldap://127.0.0.1:3891 ldap://127.0.0.1:3892
edu.vt.middleware.ldap.connectionHandler=edu.vt.middleware.ldap.handler.DefaultConnectionHandler
{{connectionStrategy=ACTIVE_PASSIVE}}
# edu.vt.middleware.ldap.connectionHandler=edu.vt.middleware.ldap.handler.DefaultConnectionHandler
{{connectionStrategy=ROUND_ROBIN}}
```

Because multiple directories are provisioned, this example is different than those which provision a single target. In the examples which provision a single ldap target, a single ldap connection is used to lookup Grouper subjects as well as to provision. In this example, an ldap connection is created to provision each target.

The connections to the three provisioned ldap targets are defined in `psp-services.xml` and `psp-vt-ldap-123.xml`.

All three targets are provisioned in a single thread, future work should allow for one thread per target.

examples	psp-example-grouper-to-openldap-multiple
DN structure	bushy
`hasMember`	member names
`isMemberOf`	group names
`member`	member DNs
`memberOf`	group DNs

Configuration Example : Grouper to Active Directory

A contribution from Sébastien Gagné, Université de Montréal.

examples	psp-example-grouper-to-active-directory
DN structure	bushy
`member`	member DNs

Configuration Example : LDAP to Grouper

This example provisions groups, stems, and memberships from an ldap directory to Grouper.

examples	psp-example-ldap-to-grouper

Configure LDAP Provisioning Target

The LDAP provisioning target connection is configured in `ldap.properties`.

Configure the default search base DN to match your directory :

ldap.properties

```
edu.vt.middleware.ldap.baseDn = dc=example,dc=edu
```

Configure authentication and encryption :

ldap.properties

```
edu.vt.middleware.ldap.bindDn=cn=Manager,dc=example,dc=edu
edu.vt.middleware.ldap.bindCredential=secret
```

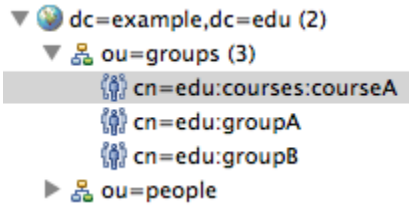
Configure the default base DN (container) for people and groups :

ldap.properties

```
# The base DN for groups.
edu.internet2.middleware.psp.groupsBaseDn = ou=groups,dc=example,dc=edu
# The base DN for people.
edu.internet2.middleware.psp.peopleBaseDn = ou=people,dc=example,dc=edu
```

Configure LDAP Provisioning Structure : Flat

In a `flat` structure all groups are provisioned under a single base DN (container ID). A `flat` group's ldap RDN is its Grouper name or displayName.



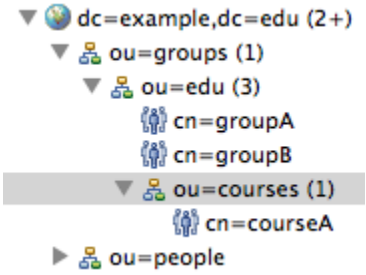
Configure the `flat` LDAP structure and `name` group RDN source attribute ID in `ldap.properties` :

ldap.properties

```
edu.internet2.middleware.psp.structure=flat
edu.internet2.middleware.psp.cnSourceAttributeID=name
```

Configure LDAP Provisioning Structure : Bushy

In a `bushy` structure groups are provisioned hierarchically, with stems as branches (ldap organizationalUnits) in the tree. A `bushy` group's RDN is its Grouper extension or displayExtension.



Configure the `bushy` LDAP structure and `extension` group RDN source attribute ID in `ldap.properties` :

ldap.properties

```
edu.internet2.middleware.psp.structure=bushy
edu.internet2.middleware.psp.cnSourceAttributeID=extension
```

Configure LDAP Base DNs

The example default base DN is `dc=example,dc=edu`. Change the base DN in `ldap.properties` :

ldap.properties

```
# The default base DN for searches.
edu.vt.middleware.ldap.baseDn=dc=example,dc=edu

# The base DN for groups.
edu.internet2.middleware.psp.groupsBaseDn=ou=groups,dc=example,dc=edu

# The base DN for people.
edu.internet2.middleware.psp.peopleBaseDn=ou=people,dc=example,dc=edu
```

as well as `sources.xml` :

sources.xml

```
<source adapterClass="edu.internet2.middleware.subject.provider.LdapSourceAdapter">
    <id>ldap</id>
    <name>LdapSourceAdapter</name>
    <type>person</type>

    <search>
        <searchType>searchSubject</searchType>
        ...
        <param>
            <param-name>base</param-name>
            <param-value>ou=people,dc=example,dc=edu</param-value>
        </param>

    </search>
    <search>
        <searchType>searchSubjectByIdentifier</searchType>
        ...
        <param>
            <param-name>base</param-name>
            <param-value>ou=people,dc=example,dc=edu</param-value>
        </param>
    </search>

    <search>
       <searchType>search</searchType>
         ...
         <param>
            <param-name>base</param-name>
            <param-value>ou=people,dc=example,dc=edu</param-value>
        </param>
    </search>
```

Configure LDAP Subject Source

Configure Grouper to look for subjects in your LDAP directory by using the ldap source adapter.

The Grouper LDAP subject source connection is defined in `sources.xml`.

sources.xml

```
<source adapterClass="edu.internet2.middleware.subject.provider.LdapSourceAdapter">
    <id>ldap</id>
    <name>LdapSourceAdapter</name>
    <type>person</type>

    <init-param>
      <param-name>ldapProperties_file</param-name>
      <param-value>ldap.properties</param-value>
    </init-param>
```

Configure LDAP Subject Source ID Other Than "ldap"

The id of the Grouper LDAP subject source adapter, `<id>ldap</id>`, appears in several psp configuration files. If your Grouper LDAP subject source adapter id is not "ldap", you should read the following and make changes to your psp configuration files. In the following examples, the Grouper LDAP subject source id has been changed from "ldap" to "ad".

In the psp service configuration `psp-services.xml`, the LDAP target to be provisioned re-uses the same pooled vt-ldap connection as the Grouper LDAP subject source adapter. The value of the vt-ldap pool id property `ldapPoolId="ldap"` should match the Grouper LDAP subject source adapter id `<id>ldap</id>` in `sources.xml`. If your Grouper LDAP subject source id is `<id>ad</ad>`, then the vt-ldap pool id should be `ldapPoolId="ad"`.

Also in the psp service configuration `psp-services.xml`, the id of the LDAP target to be provisioned, `id="ldap"`, is the SPMLv2 targetId which should match the targetId attribute of `<identifier/>` elements in the psp configuration file.

psp-services.xml

```
<Service
    id="ldap"
    xsi:type="psp-ldap-target:LdapTarget"
    ldapPoolId="ad"
    ldapPoolIdSource="grouper">
  </Service>
```

In the psp configuration file `psp.xml`, the target id of the `<identifier/>` element of objects to be provisioned, `targetId="ldap"`, should match the LDAP target id `<Service id="ldap"/>` as defined in `psp-services.xml`. If your Grouper LDAP subject source id is `<id>ad</id>`, you do NOT need to change the targetId.

```
<!-- The ladp group DN. -->
<identifier
    ref="groupDn"
    targetId="ldap"
    containerId="${edu.internet2.middleware.psp.groupsBaseDn}" />
```

In the attribute resolver configuration `psp-resolver.xml`, the Grouper LDAP subject source id, `<id>ldap</id>`, appears in several elements.

The first place that Grouper LDAP subject source id "ldap" appears in the attribute resolver configuration is in the element which defines that the `MemberDataConnector` should return the "dn" attribute for Grouper members whose subject source is "ldap". The "dn" attribute is used as the identifier of provisioned member objects. If your Grouper LDAP subject source id is `<id>ad</id>`, then the source of the "dn" attribute should be `source="ad"`.

psp-resolver.xml

```
<!-- The MemberDataConnector returns attributes representing the member whose subject id or identifier is the
principal name. -->
<resolver:DataConnector
    id="MemberDataConnector"
    xsi:type="grouper:MemberDataConnector">
    <!-- Return the "dn" attribute of members whose subject source id is "ad". -->
    <grouper:Attribute
      id="dn"
      source="ad" />
  </resolver:DataConnector>
```

The second place that "ldap" appears in the attribute resolver configuration is in the element which defines that the "id" attribute should be returned as values of the "membersLdap" attribute for Grouper members whose subject source is "ldap". The values of the "membersLdap" attribute definition, Grouper LDAP subject ids, are used to calculate group memberships. If your Grouper LDAP subject source id is `<id>ad</id>`, then the source of the "membersLdap" attribute should be `source="ad"`.

psp-resolver.xml

```
<!-- The values of the "membersLdap" attribute are the subject ids of group members from the "ldap" source. -->
  <resolver:AttributeDefinition
    id="membersLdap"
    xsi:type="grouper:Member"
    sourceAttributeID="members">
    <resolver:Dependency ref="GroupDataConnector" />
    <!-- The values of the "id" attribute are the identifiers of subjects whose source id is "ad". -->
    <grouper:Attribute
      id="id"
      source="ad" />
  </resolver:AttributeDefinition>
```

The third place that "ldap" appears in the attribute resolver configuration is in the element which defines that the "id" attribute should be returned as values of the "changeLogMembershipLdapSubjectId" attribute for Grouper members whose subject source is "ldap". The values of the "changeLogMembershipLdapSubjectId" attribute definition, Grouper LDAP subject ids, are used to calculate group memberships during processing of change log entries. If your Grouper LDAP subject source id is `<id>ad</id>`, then the sourceId should contain "ad", for example, `sourceId.getValues().contains("ad")`.

psp-resolver.xml

```
<!-- The value of the "changeLogMembershipLdapSubjectId" attribute is the subject identifier of the "ldap"
source member
    of a membership change log entry. -->
  <resolver:AttributeDefinition
    id="changeLogMembershipLdapSubjectId"
    xsi:type="ad:Script">
    <resolver:Dependency ref="AddMembershipChangeLogDataConnector" />
    <resolver:Dependency ref="DeleteMembershipChangeLogDataConnector" />
    <ad:Script><![CDATA[here|Grouper:Notifications (change log)]]></ad:Script>
  </resolver:AttributeDefinition>
```

Configure LDAP Subject Source in Grouper UI

Copy `sources.xml` and `ldap.properties` from the Grouper API to the Grouper UI.

```
cp grouper.apiBinary-2.1.0/conf/ldap.properties grouper.ui-2.1.0/dist/grouper/WEB-INF/classes/
cp grouper.apiBinary-2.1.0/conf/sources.xml grouper.ui-2.1.0/dist/grouper/WEB-INF/classes/
```

Copy `vt-ldap.jar` from the Grouper API to the Grouper UI :

```
cp grouper.apiBinary-2.1.0/lib/custom/vt-ldap-3.3.4.jar grouper.ui-2.1.0/dist/grouper/WEB-INF/lib
```


The Grouper UI will not load unless you edit `ldap.properties` in your Grouper UI installation or copy `psp-ldap-target-2.1.0-SNAPSHOT.jar` to your Grouper UI installation.

Comment out or remove the psp specific search result handlers in `ldap.properties` in your Grouper UI installation :

ldap.properties

```
# edu.vt.middleware.ldap.searchResultHandlers=edu.internet2.middleware.psp.ldap.QuotedDnResultHandler,...
```

Configure LDAP Subject Source in Grouper WS

TODO

Configure LDAP DNs Created from Grouper Names

A provisioned object requires a unique identifier. When provisioning an ldap target, provisioned object identifiers are ldap distinguished names (DNs).

The `psp-grouper-ldap` project provides a Shibboleth attribute resolver attribute definition which maps Grouper names to ldap DNs.

psp-resolver.xml

```
<!-- The LDAP DN of a group. For example, "cn=groupExtension,ou=stem,ou=groups,dc=example,dc=edu". -->
  <resolver:AttributeDefinition
    id="groupDn"
    xsi:type="psp-grouper-ldap:LdapDnFromGrouperNamePSOIdentifier"
    structure="${edu.internet2.middleware.psp.structure}"
    sourceAttributeID="groupNameInStem"
    rdnAttributeName="cn"
    baseDn="${edu.internet2.middleware.psp.groupsBaseDn}"
    baseStem="${edu.internet2.middleware.psp.baseStem}">
    <!-- Dependencies which return a "groupNameInStem" attribute whose value is the group name. -->
    <resolver:Dependency ref="groupNameInStem"/>
```

Based on a request from the University of Montreal, the Grouper base stem to be provisioned may be omitted from ldap DNs. The following table describes the affect of the base stem to be provisioned when the structure is `bushy` :

grouper base stem	grouper name	ldap dn
	edu:courses:course	cn=course,ou=courses,ou=edu,ou=groups,dc=example,dc=edu
edu	edu:courses:course	cn=course,ou=courses,ou=groups,dc=example,dc=edu

By default, the psp-example-* configuration files use the Grouper name to create ldap DNs. It is also possible to create ldap DNs from the Grouper displayName by changing the relevant sourceAttributeID to "displayName".

psp-resolver.xml

```
<resolver:AttributeDefinition
    id="groupNameInStem"
    xsi:type="grouper:FilteredName"
    sourceAttributeID="displayName">
    ...
```

Configure Grouper Stem to be Provisioned

Configure the name of the Grouper stem to be provisioned, by default this is the root stem, which is the empty string.

The Grouper stem to be provisioned is configured in `ldap.properties`.

ldap.properties

```
# The base Grouper stem to be provisioned.
edu.internet2.middleware.psp.baseStem=
```

Configure Grouper Change Log

The Grouper change log is configured in `grouper-loader.properties`.

To enable change log provisioning :

grouper-loader.properties

```
changeLog.consumer.psp.class = edu.internet2.middleware.psp.grouper.PspChangeLogConsumer
```

To schedule when the change log is processed :

grouper-loader.properties

```
changeLog.consumer.psp.quartzCron = 0 * * * * ?
```

To run full synchronizations periodically (by default every day at 5am) :

grouper-loader.properties

```
changeLog.psp.fullSync.class = edu.internet2.middleware.psp.grouper.PspChangeLogConsumer
changeLog.psp.fullSync.quartzCron = 0 0 5 * * ?
```

To run a full synchronization job at loader startup :

grouper-loader.properties

```
changeLog.psp.fullSync.runAtStartup = true
```

The Quartz cron string documentation is here.

Configure Grouper Logging

You may want to change the Grouper log file appenders in `grouper.apiBinary-2.1.0/conf/log4j.properties`.

log4j.properties

```
log4j.appender.grouper_error                            = org.apache.log4j.DailyRollingFileAppender
log4j.appender.grouper_error.File                       = ${grouper.home}logs/grouper_error.log
log4j.appender.grouper_error.DatePattern                = '.'yyyy-MM-dd

log4j.appender.grouper_event                            = org.apache.log4j.DailyRollingFileAppender
log4j.appender.grouper_event.File                       = ${grouper.home}logs/grouper_event.log
log4j.appender.grouper_event.DatePattern                = '.'yyyy-MM-dd
```

Configure Grouper Versions Prior to 2.1.0

⚠ The following changes are necessary to support provisioning Grouper API versions prior to 2.1.0 with the psp.

1. For Grouper versions prior to 2.1.0, there is a bug which will throw a NullPointerException if the following is not present in `sources.xml` :

```
<search>
    <searchType>searchSubjectByIdentifierAttributes</searchType>
    <param>
        <param-name>filter</param-name>
        <param-value>
            (&amp;(uid=%TERM%)(objectclass=person))
        </param-value>
    </param>
    <param>
        <param-name>scope</param-name>
        <param-value>SUBTREE_SCOPE</param-value>
    </param>
    <param>
        <param-name>base</param-name>
        <param-value>ou=people,dc=example,dc=edu</param-value>
    </param>
</search>
```

2. For versions prior to 2.1.0, the location of `ldap.properties` specified in `sources.xml` must be an absolute path. For versions 2.1.0 or later, the location of `ldap.properties` may be an absolute path or in your Java classpath.

For example, Grouper API version 2.0.3 requires an absolute path to ldap.properties in `sources.xml` :

```
<init-param>
    <param-name>ldapProperties_file</param-name>
    <param-value>/opt/grouper/2.0.3/grouper.apiBinary-2.0.3/conf/ldap.properties</param-value>
</init-param>
```

For Grouper UI and WS versions prior to 2.1.0, the path to `ldap.properties` specified in `sources.xml` will be different than in the Grouper API since the psp specific search result handlers must be commented out or removed in the Grouper UI :

```
<init-param>
    <param-name>ldapProperties_file</param-name>
    <param-value>/opt/grouper/2.0.3/grouper.ui-2.0.3/dist/grouper/WEB-INF/classes/ldap.properties</param-value>
</init-param>
```

3. For Grouper API versions prior to 2.1.0, the ldap source adapter in `subject.jar` does not provide the method which allows the psp to re-use the same ldap connection as the subject source. You will need to copy `lib/grouper/subject.jar` from the Grouper 2.1.0 API distribution to your pre-2.1.0 Grouper API installation.

Configure Subject API Cache

The Subject API cache is configured in `grouper.ehcache.xml`.

Adjust `maxElementsInMemory` to be greater than or equal to the number of subjects.

Adjust `timeToIdleSeconds` and `timeToLiveSeconds` ... accordingly ... ?

Some words about testing via gsh.sh and looking at cache hit/miss ratio debugging.

As of Grouper version 2.1.0, which uses Ehcache 2.4, `statistics` must be `"true"` to collect statistics which are logged at `DEBUG` level.

grouper.ehcache.xml

```
<!-- Subject resolving caching -->

 <!-- @see   CachingResolver#find(...) -->
 <cache  name="edu.internet2.middleware.grouper.subj.CachingResolver.Find"
         maxElementsInMemory="5000"
         eternal="false"
         timeToIdleSeconds="30"
         timeToLiveSeconds="120"
         overflowToDisk="false"
         statistics="true"
 />

 <!-- @see   CachingResolver#findAll(...) -->
 <cache  name="edu.internet2.middleware.grouper.subj.CachingResolver.FindAll"
         maxElementsInMemory="5000"
         eternal="false"
         timeToIdleSeconds="30"
         timeToLiveSeconds="120"
         overflowToDisk="false"
         statistics="true"
 />

 <!-- @see   CachingResolver#findByIdentifier(...) -->
 <cache  name="edu.internet2.middleware.grouper.subj.CachingResolver.FindByIdentifier"
         maxElementsInMemory="5000"
         eternal="false"
         timeToIdleSeconds="30"
         timeToLiveSeconds="120"
         overflowToDisk="false"
         statistics="true"
 />

 <!-- @see   CachingResolver#findByIdOrIdentifier(...) -->
 <cache  name="edu.internet2.middleware.grouper.subj.CachingResolver.FindByIdOrIdentifier"
         maxElementsInMemory="5000"
         eternal="false"
         timeToIdleSeconds="30"
         timeToLiveSeconds="120"
         overflowToDisk="false"
         statistics="true"
 />
```

Configure PSP : Provisioning Service Provider

The psp configuration files are :

`psp.xml`	Configuration for the objects, identifiers, attributes, and references to be provisioned to a target.
`psp-resolver.xml`	Configuration for the Shibboleth attribute resolver.
`psp-services.xml`	Configuration for Shibboleth services such as the attribute resolver, psp, and provisioning targets.
`psp-internal.xml`	Bootstraps Shibboleth.

Configure PSP : SPMLv2 Provisioned Objects, Identifiers, Attributes and References

The objects, identifiers, attributes, and references to be provisioned are defined in `psp.xml`.

Configure PSP : Provisioned Objects

Provisioned objects, or in SPMLv2 terms Provisioning Service Objects, consist of identifiers, attributes (probably), and references (maybe) to the identifiers of other objects, which are most likely located on the same provisioning target.

The following configures the psp to provision a `group` object.

psp.xml object

```
<pso
    id="group"
    authoritative="true"
    allSourceIdentifiersRef="groupNames">
</pso>
```

property	default	value
`id`		the unique id of the provisioned object
`authoritative`	`true`	If `true`, orphan objects will be deleted. Orphan objects exist on a target with no corresponding source object.
`allSourceIdentifiersRef`		The id of an attribute resolver definition whose values are all source identifiers applicable to this provisioned object.

Configure PSP : Identifiers

Identifiers consist of a string ID, a target ID, and possibly a container ID. We consider a container ID to be similar to an ldap base dn. A container ID is itself an identifier, recursing potentially indefinitely.

The following configures the psp to provision the identifier of the `grouper` object as an LDAP DN returned from the `groupDn` Shibboleth attribute definition from `psp-resolver.xml`.

psp.xml identifier

```
<pso id="group">
    <!-- The ldap group DN. -->
    <identifier
      ref="groupDn"
      targetId="ldap"
      containerId="${edu.internet2.middleware.psp.groupsBaseDn}" />
</pso>
```

property	value
`ref`	The id of the Shibboleth attribute definition whose value is an SPMLv2 PSO Identifier
`targetId`	The id of the provisioned target. Must match the id of a target configured in `psp-services.xml`
`containerId`	The string id of the pso identifier containing the object.

psp-resolver.xml identifier

```
<!-- The LDAP DN of a group. For example, "cn=group,ou=groups,dc=example,dc=edu". -->
  <resolver:AttributeDefinition
    id="groupDn"
    xsi:type="psp-grouper-ldap:LdapDnFromGrouperNamePSOIdentifier"
    structure="${edu.internet2.middleware.psp.structure}"
    sourceAttributeID="name"
    rdnAttributeName="cn"
    base="${edu.internet2.middleware.psp.groupsBaseDn}">
    <!-- Dependencies which return a "name" attribute whose value is the group name. -->
    <resolver:Dependency ref="GroupDataConnector" />
    <resolver:Dependency ref="DeleteGroupChangeLogDataConnector" />
    <resolver:Dependency ref="UpdateGroupChangeLogDataConnector" />
  </resolver:AttributeDefinition>
```

The following is an identifier expressed in SPMLv2

SPMLv2 identifier

```
<psp:pso entityName='group'>
    <psoID ID='cn=group,ou=groups,dc=example,dc=edu' targetID='ldap'/>
</psp:pso>
```

The following is a identifier expressed in LDIF

LDIF identifier

```
dn: cn=group,ou=groups,dc=example,dc=edu
```

Configure PSP : Identifying Attribute

The optional <identifyingAttribute/> of a provisioned object has two purposes : (1) to determine the schema entity of target objects returned from a lookup or search request and (2) to be converted to a query when searching a target for all identifiers. If <identifyingAttribute/> is not present, the provisioned object will be ignored during bulk requests.

psp.xml identifying attribute

```
<!-- Identifies ldap group objects which exist on the target by objectClass attribute value. -->
<identifyingAttribute
   name="objectClass"
   value="groupOfNames" />
```

(1) The provisioning service provider needs to map provisioned object identifiers to provisioned objects (schema entities).

For example, given a lookup request for the id "edu", the psp needs to know if "edu" is a group or a stem. Given the following configuration, if the provisioned object with id "edu" has an "objectclass" attribute with value "organizationalUnit", then the schema entity is "stem". If the provisioned object with id "edu" has an "objectclass" attribute with value "groupOfNames", then the schema entity is "group".

The psp evaluates all <identifyingAttribute/> elements, only one should match, otherwise an exception is thrown.

psp.xml identifying attribute

```
<pso id="stem">
  <!-- The ldap organizational unit DN. -->
  <identifier
    ref="stemDn"
    targetId="ldap"
    containerId="${edu.internet2.middleware.psp.groupsBaseDn}" />

  <!-- Identifies stem objects which exist on the target by objectclass attribute value. -->
  <identifyingAttribute
    name="objectclass"
    value="organizationalUnit" />
</pso>

<pso id="group">
  <!-- The ldap group DN. -->
  <identifier
    ref="groupDn"
    targetId="ldap"
    containerId="${edu.internet2.middleware.psp.groupsBaseDn}" />

  <!-- Identifies stem objects which exist on the target by objectclass attribute value. -->
  <identifyingAttribute
    name="objectclass"
    value="groupOfNames" />
</pso>
```

(2) In order to synchronize all objects during bulk[Calc|Diff|Sync] requests, the psp needs to know the identifiers of all provisioned objects (schema entities) on a target for which the psp is authoritative. The psp uses `<identifyingAttribute/>` and `<identifier containerId="..."/>` elements to create SPMLv2 search requests.

For example, in the configuration example above, the psp will perform an ldap search with filter "(objectClass=organizationalUnit)" to retrieve the identifiers of all provisioned stems as well as an ldap search with filter "(objectclass=groupOfNames)" to retrieve the identifiers of all provisioned groups. The base of each search will be the containerId of the `<identifier/>` element.

Configure PSP : Alternate Identifier

The optional `<alternateIdentifier/>` element configures the psp to rename provisioned objects. It refers to an attribute resolver definition whose value is the previous (old) identifier of an object after it has been renamed. If `<alternateIdentifier/>` is not present, provisioned objects will not be renamed, instead the old object will be deleted and a new object created.

psp.xml alternate identifier

```
<!-- The "old" ldap group DN calculated from group update change log events. -->
<alternateIdentifier ref="groupDnAlternateChangeLog" />
```

property	value
`ref`	The id of the Shibboleth attribute definition whose value is the previous SPMLv2 PSO Identifier.

Configure PSP : Attributes

Name value pairs. Probably multi-valued. Case sensitive names and values. We return values in the same order as they were given to us.

The following configures the psp to provision the `cn` attribute of a `group`. The value of the `cn` attribute is returned by the `cn` Shibboleth attribute definition from `psp-resolver.xml`.

psp.xml attribute

```
<pso id="group">
    <attribute name="cn" />
</pso>
```

psp-resolver.xml attribute

```
<resolver:AttributeDefinition
    id="cn"
    xsi:type="ad:Simple"
    sourceAttributeID="cn">
    <resolver:Dependency ref="GroupDataConnector" />
  </resolver:AttributeDefinition>
```

The following is an attribute expressed in SPMLv2

SPMLv2 attribute

```
<psp:pso entityName='group'>
    <psoID ID='cn=group,ou=groups,dc=example,dc=edu' targetID='ldap'/>
    <data>
        <dsml:attr xmlns:dsml='urn:oasis:names:tc:DSML:2:0:core' name='cn'>
            <dsml:value>group</dsml:value>
        </dsml:attr>
    </data>
</psp:pso>
```

The following is a attribute expressed in LDIF

LDIF attribute

```
dn: cn=group,ou=groups,dc=example,dc=edu
cn: group
```

Configure PSP : References

A reference refers to the identifier of another object. It consists of two identifiers, the "from object" and the "to object". A node in a directed graph. Directional.

The following configures the psp to provision a reference from a `group` to a `member` as values of the `member` attribute. The values of the reference are returned by the `membersLdap` and `membersGsa` Shibboleth attribute definitions from `psp-resolver.xml`.

psp.xml reference

```
<pso id="group">
    <references name="member">
        <reference
            ref="membersLdap"
            toObject="member" />
        <reference
           ref="membersGsa"
           toObject="group" />
    </references>
</pso>
```

psp-resolver.xml reference

```
<!-- The values of the "membersLdap" attribute are the subject ids of group members from the "ldap" source. -->
  <resolver:AttributeDefinition
    id="membersLdap"
    xsi:type="grouper:Member"
    sourceAttributeID="members">
    <resolver:Dependency ref="GroupDataConnector" />
    <!-- The values of the "id" attribute are the identifiers of subjects whose source id is "ldap". -->
    <grouper:Attribute
      id="id"
      source="ldap" />
  </resolver:AttributeDefinition>

  <!-- The values of the "membersGsa" attribute are the names of group members which are grouper groups. -->
  <resolver:AttributeDefinition
    id="membersGsa"
    xsi:type="grouper:Member"
    sourceAttributeID="members">
    <resolver:Dependency ref="GroupDataConnector" />
    <!-- The values of the "name" attribute are the names of groups whose source is "g:gsa". -->
    <grouper:Attribute
      id="name"
      source="g:gsa" />
  </resolver:AttributeDefinition>
```

The following is a reference expressed in SPMLv2

SPMLv2 reference

```
<psp:pso entityName='group'>
    <psoID ID='cn=group,ou=groups,dc=example,dc=edu' targetID='ldap'/>
    <capabilityData mustUnderstand='true' capabilityURI='urn:oasis:names:tc:SPML:2:0:reference'>
      <spmlref:reference xmlns='urn:oasis:names:tc:SPML:2:0' xmlns:spmlref='urn:oasis:names:tc:SPML:2:0:
reference' typeOfReference='member'>
        <spmlref:toPsoID ID='uid=123,ou=people,dc=example,dc=edu' targetID='ldap'/>
      </spmlref:reference>
      <spmlref:reference xmlns='urn:oasis:names:tc:SPML:2:0' xmlns:spmlref='urn:oasis:names:tc:SPML:2:0:
reference' typeOfReference='memberOf'>
        <spmlref:toPsoID ID='cn=group,ou=groups,dc=example,dc=edu' targetID='ldap'/>
      </spmlref:reference>
    </capabilityData>
</psp:pso>
```

The following is a reference expressed in LDIF

LDIF reference

```
dn: cn=group,ou=groups,dc=example,dc=edu
cn: group
member: uid=person,ou=people,dc=example,dc=edu

dn: uid=person,ou=people,dc=example,dc=edu
...
```

Configure PSP : Attribute Resolver

The values of the identifiers, attributes, and references to be provisioned are defined by a Shibboleth attribute resolver configuration `psp-resolver.xml`.

psp-services.xml

```
<!-- The attribute resolver. -->
  <Service
    id="psp.AttributeResolver"
    xsi:type="attribute-resolver:ShibbolethAttributeResolver">
    <ConfigurationResource
      file="/psp-resolver.xml"
      xsi:type="resource:ClasspathResource">
      <ResourceFilter
        xsi:type="grouper:ClasspathPropertyReplacement"
        xmlns="urn:mace:shibboleth:2.0:resource"
        propertyFile="/ldap.properties" />
    </ConfigurationResource>
  </Service>
```

Configure PSP : Attribute Resolver and Grouper Integration

Please see Grouper and Shibboleth Integration.

Configure PSP : Attribute Resolver and Grouper ChangeLog Integration

Real-time provisioning based on the Grouper change log involves the transformation of change log entries into provisioning operations.

Provisioned objects, identifiers, attributes, and references are calculated by the Shibboleth attribute resolver from change log entries.

Please see Grouper and Shibboleth Integration for more information on the change log data connector.

The psp-grouper-changelog project provides a change log consumer implementation, edu.internet2.middleware.psp.grouper.PspChangeLogConsumer. Documentation for Grouper change log consumers is here.

Configure PSP : PspChangeLogConsumer

The PspChangeLogConsumer processes change log entries which match hardcoded change log actions and categories. A future version should allow for configuration via xml or property files.

The change log events currently handled are :

addAttributeAssignValue
deleteAttributeAssignValue
addGroup
deleteGroup
updateGroup
addMembership
deleteMembership
addStem
deleteStem
updateStem

Configure PSP : Logging and Output

The psp is a Shibboleth service which is configured in `psp-services.xml`.

The psp uses slf4j, and with Grouper, log4j configured in `log4j.properties`.

log4j.properties

```
# Provisioning : PSP (version 2.1+)
log4j.logger.edu.internet2.middleware.psp = INFO

# Provisioning : vt-ldap
log4j.logger.edu.vt.middleware.ldap = INFO

# Provisioning : ldap target
# log4j.logger.edu.internet2.middleware.ldap = DEBUG

# Provisioning : Grouper plugin to Shibboleth attribute resolver
log4j.logger.edu.internet2.middleware.grouper.shibboleth = INFO
```

property	default	value
`logSpml`	`true`	If `true`, log SPML requests and responses in XML.
`writeRequests`	`false`	If `true`, write SPML requests.
`writeResponses`	`false`	If `true`, write SPML responses.
`pathToOutputFile`	`stdout`	The path to the file to which SPML requests and responses are written.

psp-services.xml

```
<!-- The provisioning service provider. -->
  <Service
    id="psp"
    xsi:type="psp:ProvisioningServiceProvider"
    depends-on="psp.AttributeAuthority"
    authority="psp.AttributeAuthority"
    logSpml="true"
    writeRequests="false"
    writeResponses="false"
    pathToOutputFile="">
    <ConfigurationResource
      file="/psp.xml"
      xsi:type="resource:ClasspathResource">
      <ResourceFilter
        xsi:type="grouper:ClasspathPropertyReplacement"
        xmlns="urn:mace:shibboleth:2.0:resource"
        propertyFile="/ldap.properties" />
    </ConfigurationResource>
  </Service>
```

Configure PSP : LDAP Target

The LDAP target to be provisioned is a Shibboleth service configured in `psp-services.xml`.

By default, the LDAP target to be provisioned re-uses the same `vt-ldap` connection as the Grouper LDAP source adapter.

property	default	value
`logSpml`	`true`	If `true`, log SPML requests and responses in XML.
`writeRequests`	`false`	If `true`, write SPML requests.
`writeResponses`	`false`	If `true`, write SPML responses.
`pathToOutputFile`	`stdout`	The path to the file to which SPML requests and responses are written.
`ldapPoolIdSource`	`grouper`	Re-use the `vt-ldap` ldap pool from the Grouper ldap source adapter.

psp-services.xml

```
<!-- The ldap target. The ldapPoolIdSource is either "grouper" or "spring". -->
  <!-- If ldapPoolIdSource is "spring", the ldapPoolId must be the id of the ldap pool bean in the vt-ldap xml
spring configuration. -->
  <!-- If ldapPoolIdSource is "grouper", the ldapPoolId must be the id of the LdapSourceAdapter in sources.xml
-->
  <Service
    id="ldap"
    xsi:type="psp-ldap-target:LdapTarget"
    logSpml="true"
    ldapPoolId="ldap"
    ldapPoolIdSource="grouper">
    <!-- A <ConfigurationResource/> is required to instantiate the <Service/>, so supply a do-nothing resource.
-->
    <ConfigurationResource
      file="/edu/internet2/middleware/psp/util/empty-bean.xml"
      xsi:type="resource:ClasspathResource" />
  </Service>
```

Provision Grouper

Before you can provision anything from Grouper to ldap or anywhere else, you will need to create the corresponding objects in Grouper using the UI, API, GSH, WS, loader, import, etc.

Provision Grouper : GSH

To calculate how a group should be provisioned :

`bin/gsh.sh -psp -calc edu:group`

To diff the current and correct provisioning of a group :

`bin/gsh.sh -psp -diff edu:group`

To provision or synchronize a group :

`bin/gsh.sh -psp -sync edu:group`

Provision Grouper : Grouper Change Log

To provision in real-time triggered by the Grouper change log, enable the psp consumer in `grouper-loader.properties`

grouper-loader.properties

```
changeLog.consumer.psp.class = edu.internet2.middleware.psp.grouper.PspChangeLogConsumer
```

and run the loader via

bin/gsh.sh -loader

Real-Time Changelog Provisioning Details - Probably More Than You Ever Want to Know

Changes to grouper are consumed by change log consumers. Grouper change log consumers are managed by the grouper loader and are configured in `grouper-loader.properties`.

If you are familiar with an ldap server audit log which logs ldif entries, the grouper changelog is similar, sort of, except that it writes rows to a table.

Grouper change log consumers process change log entries. A change log entry consists of single-valued attributes. A change log entry is a java representation of a row in the change log table `grouper_change_log_entry`.

For example, when a member is added to a group, the following is part of the psp log :

```
'ChangeLogEntry[timestamp=2012-05-31 11:59:56.321, sequence=344, category=membership, actionname=addMembership,
fieldName=members, subjectId=test.subject.1, sourceId=ldap, membershipType=flattened, groupName=edu:groupA,
...]'
```

When a change occurs in grouper, a representation of that change is written to a temporary table `grouper_change_log_entry_temp`. The grouper loader periodically (every minute) reads this table, performs magic like potentially updating the PIT (point-in-time auditing) tables, and writes to the `grouper_change_log_entry` table. It also assigns a sequential number to each action.

Change log entries written to the `grouper_change_log_entry` table are processed in order of sequence number by every change log consumer. The grouper loader persists the last sequence number successfully processed by each change log consumer.

Every minute, the grouper loader retrieves batches of 100 change log entries (`grouper_change_log_entry` rows), terminating at a maximum of 100,000 change log entries. These batches of 100 change log entries are passed to each change log consumer.

Change log consumers must extend the ChangeLogConsumerBase class :

```
package edu.internet2.middleware.grouper.changeLog;

import java.util.List;


/**
 * extend this class and register in the grouper-loader.properties to be a change log consumer
 * @author mchyzer
 *
 */
public abstract class ChangeLogConsumerBase {

  /**
   * process the change logs
   * @param changeLogEntryList  NOTE, DO NOT CHANGE OR EDIT THE OBJECTS IN THIS LIST, THEY MIGHT BE SHARED!
   * @param changeLogProcessorMetadata
   * @return which sequence number it got up to (which sequence number was the last one processed)
   */
  public abstract long processChangeLogEntries(List<ChangeLogEntry> changeLogEntryList,
      ChangeLogProcessorMetadata changeLogProcessorMetadata);

}
```

Each change log consumer must return the last sequence number processed. If a change log consumer wishes to retry the current change, it should return the current sequence number - 1.

The psp change log consumer is like any other grouper change log consumer in that it processes change log entries.

For each change log entry received, the psp change log consumer determines the change log category (for example "membership") and change log action (for example "addMembership") and either processes or ignores the change log entry. Supported change log categories and actions are coded in the psp change log consumer, take a look at the EventType enum in the PspChangeLogConsumer. To support a change log category and action that is not supported by the provided psp change log consumer, you will need to provide your own implementation, probably by extending the psp change log consumer class. For help, ask grouper-dev@internet2.ed.

Processing an "addMembership" change log entry results in a psp calc request, where the identifier (principal name) to be calculated is the change log sequence number :

```
<psp:calcRequest returnData='everything'>
  <psp:id ID='change_log_sequence_number:344'/>
</psp:calcRequest>
```

The change log data connectors know how to retrieve a change log entry from grouper via the change log sequence number, and return attributes to the attribute resolver representing the change log entry. In general, the change log data connectors convert a grouper change log entry into (shibboleth attribute authority) attributes. For example, the attribute authority will return the following attributes for an "addMembership" change log event :

```
'changeLogMembershipGroupDn' : org.openspml.v2.msg.spml.PSOIdentifier@e6acf477
'changeLogMembershipMemberDn' : org.openspml.v2.msg.spml.PSOIdentifier@97ebeb3b
'changeLogMembershipGroupName' : edu:groupA
'changeLogMembershipLdapSubjectId' : test.subject.1
```

The values of the "changeLogMembershipGroupDn" and "changeLogMembershipMemberDn" attributes are identifiers for the "groupMembership" and "memberMembership" provisioned service objects configured in `psp.xml`.

The "groupMembership" provisioned service object in `psp.xml` provisions the member attribute of a group calculated from an add or delete membership change log entry :

```
<!-- Provision a group membership triggered by the grouper change log. -->
<pso id="groupMembership">

  <!-- The ldap group DN calculated from membership change log events. -->
  <identifier
    ref="changeLogMembershipGroupDn"
    targetId="ldap"
    containerId="ou=groups,dc=example,dc=edu" />

  <!-- The ldap group "member" attribute. -->
  <references name="member">

    <reference
        ref="changeLogMembershipLdapSubjectId"
        toObject="member" />

    </references>

</pso>
```

The "memberMembership" provisioned service object in `psp.xml` provisions the memberOf attribute of a member calculated from an add or delete membership change log entry :

```
<!-- Provision a member's membership triggered by the grouper change log. -->
<pso id="memberMembership">

  <!-- The ldap group DN calculated from membership change log events. -->
  <identifier
    ref="changeLogMembershipMemberDn"
    targetId="ldap"
    containerId="ou=people,dc=example,dc=edu" />

  <!-- The ldap member "memberOf" attribute. -->
  <references name="memberOf">

    <reference
      ref="changeLogMembershipGroupName"
      toObject="group" />

  </references>

</pso>
```

The next step in processing by the psp after resolving attributes via the attribute authority is to resolve references. The values of the "changeLogMembershipLdapSubjectId" and "changeLogMembershipGroupName" attributes are converted to identifiers by executing psp calc requests.

In the following example, the psp change log consumer resolves provisioned service object identifiers for the principal with name 'edu:groupA', and returns an ldap dn 'cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' :

```
<psp:calcRequest returnData='identifier'>
  <psp:id ID='edu:groupA'/>
  <psp:schemaEntity targetID='ldap' entityName='group'/>
</psp:calcRequest>

<psp:calcResponse>
  <psp:id ID='edu:groupA'/>
  <psp:pso entityName='group'>
    <psoID ID='cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' targetID='ldap'/>
  </psp:pso>
</psp:calcResponse>
```

In the following example, the psp change log consumer resolves provisioned service object identifiers for the principal with name 'test.subject.1', and returns an ldap dn 'uid=test.subject.1,ou=people,dc=example,dc=edu' :

```
<psp:calcRequest returnData='identifier'>
  <psp:id ID='test.subject.1'/>
  <psp:schemaEntity targetID='ldap' entityName='member'/>
</psp:calcRequest>

<psp:calcResponse >
  <psp:id ID='test.subject.1'/>
  <psp:pso entityName='member'>
    <psoID ID='uid=test.subject.1,ou=people,dc=example,dc=edu' targetID='ldap'/>
  </psp:pso>
</psp:calcResponse>
```

When provisioning ldap directories, the member dn is either retrieved via an ldap grouper subject lookup or via the grouper subject source cache.

After resolving reference identifiers, a calc response is finally returned by the psp for the calc request whose id is the change log sequence number. The calc response returned by the psp represents how an add or delete membership change log entry should be provisioned, for example :

```
<psp:calcRequest returnData='everything'>
  <psp:id ID='change_log_sequence_number:344'/>
</psp:calcRequest>

<psp:calcResponse >
  <psp:id ID='change_log_sequence_number:344'/>
  <psp:pso entityName='groupMembership'>
    <psoID ID='cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' targetID='ldap'/>
    <capabilityData ... >
      <spmlref:reference typeOfReference='member' ...>
        <spmlref:toPsoID ID='uid=test.subject.1,ou=people,dc=example,dc=edu' targetID='ldap'/>
      </spmlref:reference>
    </capabilityData>
  </psp:pso>
  <psp:pso entityName='memberMembership'>
    <psoID ID='uid=test.subject.1,ou=people,dc=example,dc=edu' targetID='ldap'/>
    <capabilityData ... >
      <spmlref:reference typeOfReference='memberOf' ... >
        <spmlref:toPsoID ID='cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' targetID='ldap'/>
      </spmlref:reference>
    </capabilityData>
  </psp:pso>
</psp:calcResponse>
```

The next step in change log entry processing by the psp change log consumer is to determine the difference between how each object should be provisioned (the calc response) and how each object is currently provisioned.

For example, for each reference that should or should not be provisioned for an add or delete membership change log entry, the psp change log consumer performs an spml 'hasReference' search. Of course, the spml 'hasReference' search must be supported by each provisioned target implementation.

In the following example, the object with identifier 'cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' is queried for whether or not it has a 'member' reference to 'uid=test.subject.1,ou=people,dc=example,dc=edu' :

```
<spmlsearch:SearchRequest returnData='identifier' ... >
  <spmlsearch:query targetID='ldap' scope='pso'>
    <spmlref:hasReference typeOfReference='member'>
      <spmlref:toPsoID ID='uid=test.subject.1,ou=people,dc=example,dc=edu' targetID='ldap'/>
    </spmlref:hasReference>
    <spmlsearch:basePsoID ID='cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' targetID='ldap'/>
  </spmlsearch:query>
</spmlsearch:SearchRequest>

<spmlsearch:searchResponse status='success' ... />
```

The search response is successful but does not contain any provisioned service objects, which means that the reference does not exist.

Because the reference that should exist does not, a modify request is executed by the psp to add a 'member' reference with id 'uid=test.subject.1, ou=people,dc=example,dc=edu' to the object with id 'cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' :

```
<modifyRequest entityName='groupMembership' returnData='identifier' ... >
  <psoID ID='cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' targetID='ldap'/>
  <modification modificationMode='add'>
    <capabilityData ... >
      <spmlref:reference typeOfReference='member' ... >
        <spmlref:toPsoID ID='uid=test.subject.1,ou=people,dc=example,dc=edu' targetID='ldap'/>
      </spmlref:reference>
    </capabilityData>
  </modification>
</modifyRequest>

<modifyResponse status='success' ... >
  <pso>
    <psoID ID='cn=groupA,ou=edu,ou=groups,dc=example,dc=edu' targetID='ldap'/>
  </pso>
</modifyResponse>
```

Phew, done. In the above example, the psp has added a member to a group as a result of processing an add membership change log entry. It will also add the memberOf attribute to the member object, but I have omitted that xml.

For more detailed examples, take a look at the spml requests and responses in `src/test/resources` of the various `psp-example-grouper-to-ldap` projects.

Real-Time Provisioning Beta-Testing : Grouper Subject Sources

Institution	Subject Source	Number of Subjects	Subject ID
LIGO	LDAP	1,000	dn: employeeNumber=882,ou=people,dc=ligo,dc=org
Penn State	LDAP	165,000	dn:uid=xyx123,dc=psu,dc=edu
UCLA	LDAP	40,000	
UMontreal	LDAP	120,000	sAMAccountName (value same as cn)
UVienna	Undecided	155,000	cn, uid
UWMadison			

Real-Time Provisioning Beta-Testing : Provisioning Targets

Institution	Target	Implementation
LIGO	LDAP	OpenLDAP 2.4.x
Penn State	LDAP	IBM Tivoli Directory Server
UCLA	LDAP	Sun Java System Directory Server Enterprise Edition 6.3.1
UMontreal	LDAP	Active Directory
UVienna	LDAP	Active Directory, OpenLDAP
UWMadison		

Real-Time Provisioning Beta-Testing : Provisioning memberOf

The groups that a member *is a member of* may be provisioned to the memberOf attribute. Some LDAP implementations, such as Active Directory, automatically maintain the memberOf attribute. OpenLDAP maintains the memberOf attribute automatically via the memberOf overlay. The value of the memberOf attribute is typically a group DN.

Institution	memberOf for members (people)	memberOf for groups
LIGO	+	
Penn State	+	
UCLA	-	
UMontreal	automatic (Active Directory)	automatic (Active Directory)
UVienna	automatic (Active Directory), OpenLDAP+memberOf	automatic (Active Directory), OpenLDAP+memberOf
UWMadison	+	

Real-Time Provisioning Beta-Testing : Provisioning eduMember

The eduMember objectClass defines the isMemberOf and hasMember attributes, whose values are identifiers which are not DNs.

Institution	isMemberOf	hasMember
LIGO	+	+
Penn State	+	+
UCLA	uclaIsMemberOf	uclaHasMember
UMontreal	-	-
UVienna	-	-
UWMadison	+	+

Real-Time Provisioning Beta-Testing : Provisioning eduCourse

The eduCourse objectClass defines course related attributes.

Institution	eduCourse
LIGO	-
Penn State	-
UCLA	-
UMontreal	-
UVienna	-
UWMadison	+

Real-Time Provisioning Beta-Testing : Provisioning Structure

The group provisioning structure may be either `flat` or `bushy`. A `flat` structure provisions all groups into a single container. A `bushy` structure provisions groups hierarchically.

For example, the DN of a group with name 'edu:stem:group' in a `flat` structure looks like :

dn: cn=edu:stem:group,ou=groups,dc=example,dc=edu

while the DN of a group with name 'edu:stem:group' in a `bushy` structure looks like :

dn: cn=group,ou=stem,ou=edu,ou=groups,dc=example,dc=edu

Institution	Structure (flat or bushy)
LIGO	bushy
Penn State	flat

UCLA	flat
UMontreal	bushy
UVienna	?
UWMadison	flat

Real-Time Provisioning Beta-Testing : Membership Structure

Given groupA with memberA and groupB with memberB :

```
dn : cn=groupA,ou=groups
member: cn=memberA,ou=people

dn: cn=groupB,ou=groups
member: cn=memberB,ou=people
```

If groupB is added as a member to groupA, how do you want groupA to be provisioned :

everything :

```
dn : cn=groupA,ou=groups
member: cn=memberA,ou=people
member: cn=memberB,ou=people
member: cn=groupB,ou=people
```

immediate :

```
dn : cn=groupA,ou=groups
member: cn=memberA,ou=people
member: cn=groupB,ou=people
```

The everything membership structure handles applications which may not support nested groups and represents the nested structure of the group memberships.

The same membership structure applies to memberOf :

everything :

```
dn: cn=memberB,ou=people
memberOf: cn=groupB,ou=groups
memberOf: cn=groupA,ou=groups
```

immediate :

```
dn: cn=memberB,ou=people
memberOf: cn=groupB,ou=groups
```

```

```

Institution	member	memberOf
LIGO	everything	everything
Penn State		
UCLA		
UMontreal	immediate	immediate
UVienna	everything	everything
UWMadison		

Excluding LDAP provisioning for groups based on group name

This is based on a contribution by NYU which has been integrated into the Grouper API - Selective Group Exclusion When Provisioning to LDAP

1. In grouper.properties, set:

```
hooks.group.class=edu.internet2.middleware.grouper.hooks.examples.LDAPProvisioningHook
```

2. Also, in grouper.properties, add the names that you want to exclude (regular expressions):

```
LDAPProvisioningHook.exclude.regex.0=.*_excludes$
LDAPProvisioningHook.exclude.regex.1=.*_includes$
LDAPProvisioningHook.exclude.regex.2=.*_systemOfRecord$
LDAPProvisioningHook.exclude.regex.3=.*_systemOfRecordAndIncludes$
```

3. In the psp-resolver.xml, in each section (there are a few) that returns groups, subtract the following:

```
<grouper:Filter xsi:type="grouper:GroupExactAttribute" name="LDAPProvisioningExclude" value="true" />
```

So for example, the following:

```
   <grouper:Filter xsi:type="grouper:MINUS">
      <!-- The GroupInStem filter matches groups which are children of the given stem. -->
      <grouper:Filter
        xsi:type="grouper:GroupInStem"
        name="${edu.internet2.middleware.psp.baseStem}"
        scope="SUB" />
      <grouper:Filter
        xsi:type="grouper:GroupInStem"
        name="etc"
        scope="SUB" />
    </grouper:Filter>
```

.. would instead become:

```
<grouper:Filter xsi:type="grouper:MINUS">
    <grouper:Filter xsi:type="grouper:MINUS">
      <!-- The GroupInStem filter matches groups which are children of the given stem. -->
      <grouper:Filter
        xsi:type="grouper:GroupInStem"
        name="${edu.internet2.middleware.psp.baseStem}"
        scope="SUB" />
      <grouper:Filter
        xsi:type="grouper:GroupInStem"
        name="etc"
        scope="SUB" />
    </grouper:Filter>
    <grouper:Filter xsi:type="grouper:GroupExactAttribute" name="LDAPProvisioningExclude" value="true" />
 </grouper:Filter>
```

See Also

Directions for Provisioning Strategy