Grouper Provisioning: PSPNG (Legacy)

There is a new Grouper Provisioning Framework, released with Grouper 2.6 and above.
This page on PSPNG is no longer current as of 2022.

Provisioning's job is to reflect groups and their memberships in other systems. This is handled in Grouper via the PSPNG component, which is mostly a client of the Grouper changelog and is designed as a ChangeLog Consumer (CLC). Detected changes in Grouper via the changelog are picked by PSPNG and evaluated for provisioning operations selectively.

- History
- Development Status and Features
- Configuration
  - PSPNG JEXL object notations
  - LDAP Properties
- Advanced Ldaptive Properties
- Run PSPNG
- Group/Folder Selection
  - How do I create PSPNG attribute definitions?
  - How do I assign PSPNG attribute definitions?
- Logging & Troubleshooting
- Account creation
- Full SYNC Provisioning
- PSPNG Debugging
- Provisioning "empty" groupOfNames groups
- Provisioner Templates
  - Bushy DNs
  - GROUP OF UNIQUE NAMES
  - POSIX GROUPS
  - ACTIVE DIRECTORY GROUPS
  - USER ATTRIBUTES

History

Over the years, dozens of provisioners have been created -- some focused on a single destination type and others with some generic functionality combined with a very wide variety of options and capabilities.

Starting in 2014, the Grouper Team and Users concluded that the provisioning priorities should change: less flexibility and increased simplicity and performance. The Next Generation of the Grouper Provisioning Service Provider (aka PSPNG) was defined in Post PSP Provisioning .



As of Grouper 2.6 and above the PSPNG is no longer used. Please see Grouper Provisioning Framework documentation

Development Status and Features

PSPNG's general structure should be ready to provision various targets, but its current implementation is limited to provisioning LDAP targets like:

- -(Unix) LDAP Groups: (GroupOfUniqueNames, GroupOfNames, PosixGroup)
- -Active Directory Groups
- -LDAP Attributes (like eduPersonEntitlement)

Configuration

PSPNG's Configuration is done via the conf/grouper-loader.properties file, in the grouper API Binary, with a paragraph for each provisioning destination, as well as an additional paragraph that enables and configures FullSync operation. There are several configuration options documented in the following spreadsheet:

Provisioner Type	Parameter	Default	Description	Default Behavior
All Provisioners	provisionerName	<required>		
	enabled	true	Whether the provisioner will make changes. When this is false, the provisioner skips all changelog or full-sync requests.	The provisioner reflects changes in Grouper downstream to the target system.

	groupSelectionExpression	${
		
		utils.containedWithin(provisionerName,
		
		stemAttributes['etc:pspng:provision_to'],
		
		groupAttributes['etc:pspng:provision_to'])
		
		&&
		!utils.containedWithin(provisionerName,
		
		stemAttributes['etc:pspng:do_not_provision_to'],
		
		groupAttributes['etc:pspng:do_not_provision_to'])
		
		}
			Jexl expression that refers to stem_attributes, group_attributes, or group	Provision groups if <provisionerName> is in a group or stem provision_to attribute AND NOT in a do_not_provision_to attribute

	attributesUsedInGroupSelectionExpression	etc:pspng: provision_to, etc:pspng: do_not_provision_to	List of Grouper Attributes used in group selection. Used to improve the speed of finding relevant groups.	Use the groupSelectionExpression against groups either with provision_to /do_not_provision_to Attributes or in folders with those attributes.
	attributesUsedInGroupSelectionExpressionAreComparedToProvisionerName	true	True: the values of the attributes listed in attributesUsedInGroupSelectionExpression are the provisioner names. False: The attributes listed in attributesUsedInGroupSelectionExpression have different values than the provisioner.	PSPNG does a database search for Attribute=<provisioner name> which narrows the number of groups to compare to the groupSelectionExpression.
	grouperDataCacheTime_secs	600 (seconds)	How long should Grouper (Group, Stem, Subject) data be cached by the provisioners?	Grouper data will be cached for 10 minutes, though it is flushed when groups change.
	grouperGroupCacheSize	10000 (Groups)	How many Grouper Groups should be kept in memory at a time?	
	grouperSubjectCacheSize	10000 (Subjects)	How many Grouper Subjects should be kept in memory at a time?	
	needsTargetSystemUsers	FALSE	Does provisioniner need User/Subject information from the Target System? For example, do any JEXL expressions need information that is not available in Grouper Subjects?	All provisioning will be done based (only) on Grouper-Subject information.
	needsTargetSystemGroups	FALSE	Does provisioner need group information from the Target System? For example, this information could be used in various JEXL expressions.	All provisioning will need to be done based on Grouper-Group information
	createMissingUsers	FALSE	Only used when needsTargetSystemUsers=true: Should users be created when they cannot be found?	Users will not be created by Grouper Provisioning. Provisioning actions that require the users will fail.
	userSearch_batchSize	50	Only used when needsTargetSystemUsers=true: How many users can be sought in a single Fetch?	Fetches will seek information for up to 50 users at a single time.
	groupSearch_batchSize	50	Only used when needsTargetSystemGroups=true: How many groups can be sought in a single Fetch?	Fetches will seek information for up to 50 groups at a time.
	supportsEmptyGroups	TRUE	Can groups be created without any members? If so, it is easier to create them separately from membership changes.	Yes, create groups as soon as possible.

	`sleepTimeAfterError_ms`	`1000`	FullSync: Wait a bit before retrying a group that has failed. This prevents aggressive infinite loops.	1 second pause before retrying a failed group.
LdapProvisioner (Abstract)	`ldapPoolName`	`<required>`	What ldap pool should be used by this provisioner	
	`userSearchBaseDn`	`null`	Where to find users?	Required if provisioner needsTargetSystemUsers=true
	`userSearchFilter`	`null`	Jexl expression that refers to stem_attributes, group_attributes, or group	How to find users in the Target System?
	`userSearchAttributes[]`	`dn,cn,uid, mail, samAccountName, uidNumber, objectclass`	Comma-separated list of attributes that are useful for logging and that are needed by userSearchFilter or by a subclass's ValueFormats	Reads common attributes from either Unix or ActiveDirectory LDAP servers
	`searchResultPagingEnabled`	`TRUE`	Whether paging should be enabled in LDAP search requests; This typically requires the paging extension to be enabled and configured in LDAP to avoid LDAP Error Code 12.	TRUE
	`ldapSearchResultPagingSize`	`100`	How many result objects can be pulled by a single request. This is small to avoid problems by default.	Break the results of a large query into fairly tiny chunks.
	`ldapUserCacheTime_secs`	`600`	How long to keep User information in memory?	Keep User information in memory for 10 minutes, though user-information is flushed when users are changed by a provisioner
	`ldapUserCacheSize`	`10000`	Deprecated: use the more general targetSystemUserCacheSize. How many LDAP accounts can be kept in memory at a time, indexed by the Subject mapped to them?	Keep the last 10000 users found by searching with Subject information
	`isActiveDirectory`	`FALSE`	Is this an active-directory server? If so, then AD-specific attribute-value-paging is enabled. Also, member (reverse user-to-group virtual attribute) is enabled.	LDAP server is treated like a non-active-directory server. Problems will occur with full-sync of large groups.
	`maxValuesToChangePerOperation`	`100`	How many values can be added/removed from an attribute in a single ldap operation	Breaks large list of values that need to be added/removed from an attribute into chunks that this size. For example, 5000 values that need to be added would be added in 50 chunks of 100 values each.
	targetSystemUserCacheSize	10000		
	`userCreationBaseDn`	null	Warning: Grouper PSPNG is not a good provisioner for Accounts/Subjects. See 'Account Creation' section below. Where should account-/subject-objects be created when they don't already exist? This is only used if createMissingUsers=true	This is appended to the dn attribute produced by the userCreationLdifTemplate.
	userCreationLdifTemplate	null	Warning: Grouper PSPNG is not a good provisioner for Accounts/Subjects. See 'Account Creation' section below. What account/subject ldap objects should be created when they don't already exist in the LDAP directory? This is only used if createMissingUsers=true	

LdapGroupProvisioner (Also Provisioner and LDAPProvisioner)	`memberAttributeName`	`'member' for AD`
		
		`<required> otherwise`
			What attribute represents a group's members in the Target System?	Active Directory should just work. Otherwise, this is required.

	`memberAttributeValueFormat`	`${ldapUser.dn}`	What value (typically based on Subject or TargetSystemUser information) is written into the memberAttributeName attribute of groups?	Active Directory and GroupOfUniqueNames will typically work. This is a JEXL expression and is parsed at runtime. You may choose to script your way into this, by perhaps choosing a specific attribute: `${ldapUser.getStringValue("uid")}`

	`groupAttributeName`	`memberof for AD`
		
		`null otherwise`
			Virtual attribute of accounts that lists their groups	

	groupCreationLdifTemplate	null	What LDIF should be written to the directory to add a group. Multiple lines need to be separated by || (double-pipes). The DN of the LDIF will be combined with groupCreationBaseDn>	For AD, limit to less than 1024 if sending description, like this: `description: ${org.apache.commons.lang3.StringUtils.abbreviate(group.description == null ? null : group.description.replaceAll("\\r\\n"," ").replaceAll("\\n"," ").replaceAll("\\r"," "), 900)}`
	groupCreationBaseDn	<groupSearchBaseDn>	Where should groups be created? At group-creation time, this is appended to the DN that results from the groupCreationLdifTemplate.	Groups are created starting at the top of the search BaseDn.
	groupSearchBaseDn	<required>	Where are groups found?	
	grouperIsAuthoritative	FALSE	If set to TRUE, groups under groupCreationBaseDn that are not in Grouper will be removed at the end of a full sync.	During full syncs, groups are not removed if they do not match the allGroupsSearchFilter or groupSelectionExpression.
	allGroupsSearchFilter	null	FUTURE: How to find all the groups that grouper-provisioning maintains. If <grouperIsAuthoritative>, then groups found via this filter will be removed during a full sync.	Groups are not removed when they are removed from Grouper nor when they no longer match the groupSelectionExpression.
	singleGroupSearchFilter	<required>	How to find a group, based on Grouper Group (or stem) information	
	groupSearchAttributes	cn, gidNumber, samAccountName, objectclass	Attributes that should be read from groups when searching for them. This needs to include all the attributes used in singleGroupSearchFilter. This should not include the attribute which holds the group's members.	Support common, basic singleGroupSearchFilters.
	ldapGroupCacheTime_secs	600	How long should LDAP-Group information be cached in memory?	Keep LDAP Group information in memory for 10 minutes, though it is flushed when users are changed by a provisioner.
	ldapGroupCacheSize	10000	How many LDAP groups to keep in memory, indexed by Grouper Group.	
	needsTargetSystemUsers	TRUE	See above (JEXL expressions use User and Group information from the Target System)	
	needsTargetSystemGroups	TRUE	See above (JEXL expressions use User and Group information from the Target System)	
	allowEmptyDnAttributeValues	FALSE	v2.5.51+ Set true to allow attributes that require DN syntax to be empty in order to support use of the null DN (also known as the zero-length DN). By doing so the PSP-NG is able to provision an "empty" group with objectClass groupOfNames and the attribute 'member' with no value (the null DN)	false, remove the attribute with no value from the LDIF before creating the group
	removeNullDnFromGroupLdifCreationTemplate	FALSE	v2.5.51+ Set true so that LDIF from the group creation LDIF template is filtered and the membership attribute (e.g. member) with a null DN is removed before the LDIF is used to compare an existing group in the LDAP directory with the computed LDIF. This can be used with a group creation LDIF template that does include the member attribute with null DN to support the objectClass groupOfNames and provisioning of empty groups.	false, no filtering of the group creation LDIF is done to remove a membership attribute with a null DN.
LdapAttributeProvisioner (Also Provisioner and LDAPProvisioner)	provisionedAttributeName	<required>	What attribute is changed in User LDAP objects to represent group membership?	
	provisionedAttributeValueFormat	${group.name}	What value (typically based on Subject or TargetSystemUser information) is written into the provisionedAttributeName users?	The stem:Group name is written to the attribute specified in <provisionedAttributeName>
	needsTargetSystemUsers	TRUE	See above (JEXL expressions only use User information from the Target System)	
	needsTargetSystemGroups	FALSE	See above (JEXL expressions only use User information from the Target System)	
	allProvisionedValuesPrefix	null	What values of the attribute is grouper authoritative for during a full sync? null (default) or empty means that pspng will only process removals as memberships change, and won't clean up unknown attribute values.	Warning: Grouper should have full control over the target attribute to avoid complications that come from sharing attributes with multiple provisioning tools.

PSPNG JEXL object notations

JEXL Syntax	Description	Example string
`${group.name}`	This will return the full grouper "name" for the group that is part of the event.	folderA:folderB:FolderC:GroupDisplayName
`${ldapUser.getStringValue("uid")}`	This will return an attribute from the LDAP user (that is likely found via a search during the event)	Depends on the data in the connected Ldap service.

LDAP Properties

LDAP configuration is done based on the ldaptive library's property configuration. A paragraph of ldap configuration is created in **grouper-loader.properties** for each LDAP endpoint, and that paragraph is referenced by the appropriate provisioners. Comments in **grouper-loader.base.properties** refers to these "LDAP pools" as "LDAP connections".

> **Targeting Active Directory**
>
> Where Active Directory is the target environment, make sure you are pointing to a FQDN with active/standby load balancing or to a primary node. Other forms of load balancing can lead to inconsistent results or AD conflict CNF objects.

> At this time, the LDAP bindCredential cannot be encrypted via the Grouper morphstring.

```
ldap.groupOfNames.ldapUrl = ldaps: //hostname
ldap.groupOfNames.bindDn = cn=xxxxxx,ou=xxxxx
ldap.groupOfNames.bindCredential = xxxxx
ldap.groupOfNames.someOtherLdapProperty = value

# in another paragraph a provisioner is associated with the LDAP pool name

changeLog.consumer.pspng_groupOfUniqueNames.ldapPoolName = groupOfNames
```

Advanced Ldaptive Properties

PSPNG relies on the Ldaptive library for all LDAP-related operations. To learn more about what other LDAP properties are available, one simple example can be found here. Moving into more realistic examples will probably be helped by looking at the ldaptive configuration classes and the setters available within them: connections, pooling, binding (sasl, gssapi, x509, jks, etc).

In case it is helpful, this is currently implemented here. You may also wish to take a look at GRP-1306 to learn more about the differences between vtdap (the previously used LDAP library) vs ldaptive.

Run PSPNG

PSPNG runs as part of the grouper loader. So simply run the loader with:

```
cd <grouper-api-binary-folder>/bin
./gsh -loader
```

Group/Folder Selection

Groups can be enabled and disabled based on Group or Folder/Stem information and attributes. By default, two multivalued attributes are used:

Attribute Definition	Attribute Name	Description
etc:pspng:provision_to_def	etc:pspng:provision_to	Enable downstream provisioning

etc:pspng:do_not_provision_to_def	etc:pspng:do_not_provision_to	Disable downstream provisioning

The groupSelectionExpression can be modified to look at different group characteristics or group/folder attributes. If you replace the default provision_to /do_not_provision_to expression with **an expression that does not reference Grouper Attributes**, you'll need to still put provision_to on the parent folder (s) of the groups you wish to provision. GRP-1903 will fix this so you'll be able to set attributesUsedInGroupSelectionExpression to an empty value. If you use an expression that uses other Attributes, then make sure attributesUsedInGroupSelectionExpression is a comma-separated list of the attributes you need.

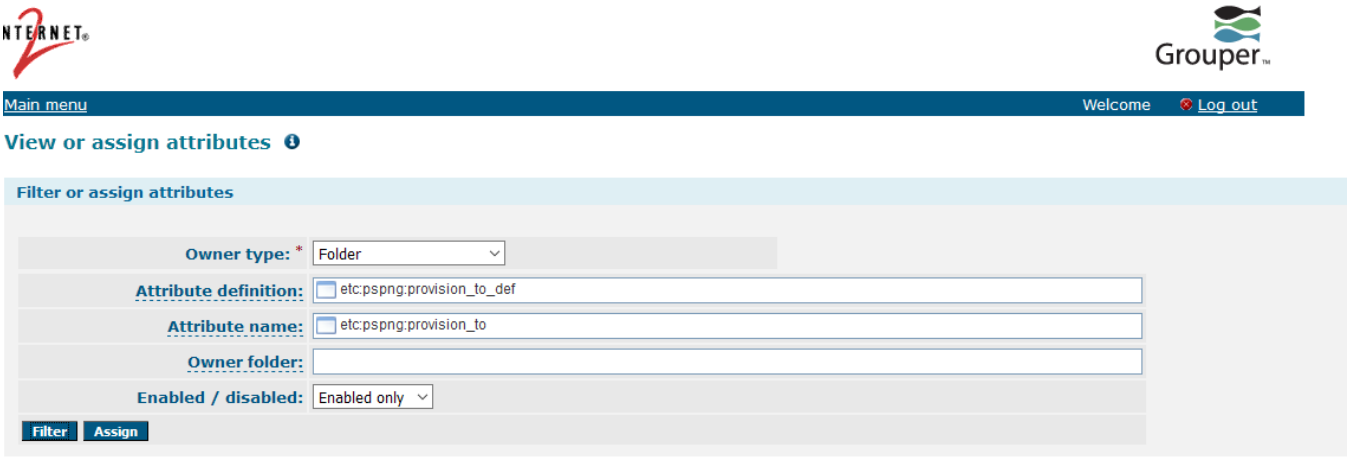
How do I create PSPNG attribute definitions?

These attribute definitions are auto-created by Grouper the very first time PSPNG runs. If you have configured PSPNG to run on startup, then the attributes will be created then or else you may need to wait for the scheduled PSPNG job to kick in.

If you need to, you can always create these via the Grouper Shell.

How do I assign PSPNG attribute definitions?

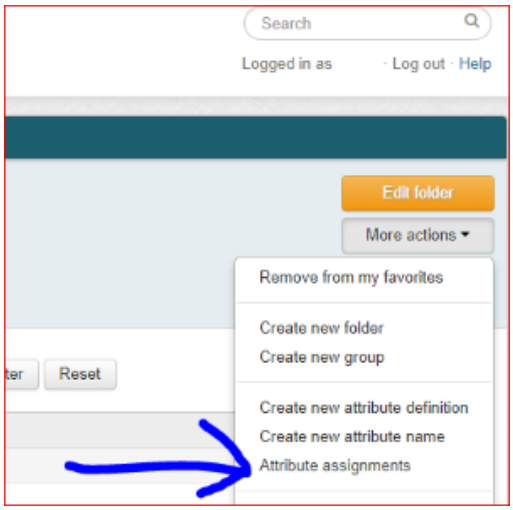
These attributes need to be assigned to Groups or Folders via the Lite UI.



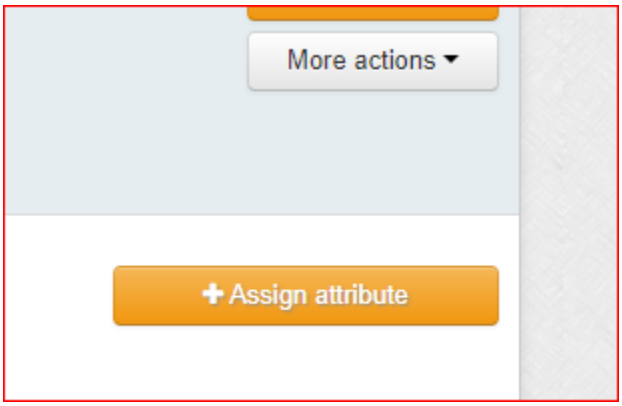
As of Grouper v2.4:

These attributes need to be assigned to Groups or Folders via the UI. (After configuring grouper-loader.properties and adding the PSPNG attribute definitions)

- Navigate to the folder or group in the UI.
  - More actions button (upper right corner of the working area)
  - Attribute assignments (menu item)



- Click the “+ Assign attribute” button



- Add the attribute that you want based on if you want to provision, or not provision.
  - etc:pspng:provision_to

    OR

  - etc:pspng:do_not_provision_to
  - clck the “save” button

- After the attribute is assigned to a folder or group, you need to assign it a value.

The value of the attribute **MUST** match the provisioner name that is defined in the grouper-loader.properties configuration (i.e. pspng_groupOfUniqueNames).

- Add a value to the attribute that matches the grouper-loader.properties configuration you want to use(or not use)
  - Click on the “Actions” button at the end of the attribute assignment row.
  - Select “Add value”.Enter (type) the value you want in the ‘Value to add’ field.
  - Click the “Submit” button.

NOTE: PSPNG will evaluate whether a group or stem/folder qualifies for provisioning by running the group-selecting filter as a JEXL expression. The expression is able to process and evaluate attribute definitions on a given group/folder and returns true/false to indicate whether PSPNG should continue with downstream provisioning.

Logging & Troubleshooting

To troubleshoot PSPNG configuration, you may need to add the following logging configuration to the log4j.properties file:

```
log4j.logger.edu.internet2.middleware.grouper.pspng=DEBUG
log4j.logger.edu.internet2.middleware.grouper.changeLog=DEBUG
```

Account creation

To facilitate (load) testing, Grouper PSPNG has limited ability to create accounts in a target system. This ability is very rudimentary and should not be used to maintain accounts in production; such maintenance should be done by an IdM product or some other ETL mechanism. If you choose to ignore this best practice, you'll probably run into two main problems:

1. Grouper/PSPNG has no ability to keep accounts up to date; it only creates them if they're missing, and
2. Your Grouper-subject mappings probably don't have some of the information needed for account provisioning. And, even if all you need is name, username, and email address (which are probably in your subject mappings), you'll still run into problem (1) where Grouper offers no mechanism to update name and email address when they change in your subject source.

Full SYNC Provisioning

Upgrade warning (PSPNG Patch #11, June 8, 2017)

The Full-Sync-provisioning improvements in Grouper 2.3/PSPNG Patch #11 are configured with new, better properties. (You should *no longer use* the properties that start with changeLog.psp.fullSync.)

In the background of PSPNG, there is always a full-sync-provisioning engine running which is automatically used when incremental provisioning finds conflicting changes or otherwise is unable to handle the changelog events. The full-sync items in grouper-loader.properties do not alter/configure the background engine but instead define quartz jobs that send all the groups marked for provisioning into the queues that drive the full-sync engine. Additionally, if a provisioner is "authoritative" (changeLog.consumer.<provisioner>.grouperIsAuthoritative=true), then a cleanup task is included in the scheduled job which will delete extra groups or attributes that no longer exist in the Group Registry. (Note: Data in a provisioning target will always get deleted by ***incremental*** provisioning when groups are deleted, regardless whether a provisioner is authoritative. Full-sync cleanup is just a safety net in case incremental provisioning misses something, or if the data was written into the target system outside of PSPNG.)

Each provisioner that wants to schedule periodic full-syncs will need the following lines included in grouper-loader.properties:

Full Sync Provisioner

```
otherJob.<provisioner-name>_full.class = edu.internet2.middleware.grouper.pspng.FullSyncStarter
otherJob.<provisioner-name>_full.quartzCron = 0 0 0 * * ?   #Every midnight

# Note, there is presently no "runAtStartup" option as there was before PSPNG Patch #11. Please see GRP-1563.
```

<provisioner-name> is the 3rd field of the properties used to configure the rest of the provisioner. For instance, <provisioner-name> is xyz for the following config snippet:

```
changeLog.consumer.xyz.class = edu.internet2.middleware.grouper.pspng.PspChangelogConsumerShim
changeLog.consumer.xyz.type = edu.internet2.middleware.grouper.pspng.LdapGroupProvisioner
...
```

and you would then create

otherJob.xyz_full.class = edu.internet2.middleware.grouper.pspng.FullSyncStarter

otherJob.xyz_full.quartzCron = 0 0 0 * * ?

PSPNG Debugging

Group selection can be evaluated with the following (interactive) gsh statements. (Grouper 2.3/PSPNG Patch #21 and later)

Full Sync Provisioner

```
provisioner_name="xyz"; // Whatever your provisioner is called in grouper_loader.properties
gs=GrouperSession.startRootSession();
provisioner=edu.internet2.middleware.grouper.pspng.ProvisionerFactory.createProvisioner(provisioner_name,false);
provisioner.getAllGroupsForProvisioner();
```

Full sync of all provisioned groups can be executed:

Versions less than PSPNG 2.4.0 patch #7

Full Sync Provisioner

```
provisioner_name="xyz"; // Whatever your provisioner is called in grouper_loader.properties
gs=GrouperSession.startRootSession();
full_provisioner=edu.internet2.middleware.grouper.pspng.FullSyncProvisionerFactory.getFullSyncer
(provisioner_name);
full_provisioner.startFullSyncOfAllGroupsAndWaitForCompletion()
```

Versions after PSPNG 2.4.0 patch #8

```
provisioner_name="xyz"; // Whatever your provisioner is called in grouper_loader.properties
gs=GrouperSession.startRootSession();
full_provisioner=edu.internet2.middleware.grouper.pspng.FullSyncProvisionerFactory.getFullSyncer
(provisioner_name);
edu.internet2.middleware.grouper.app.loader.db.Hib3GrouperLoaderLog hib3LoaderLog = new edu.internet2.
middleware.grouper.app.loader.db.Hib3GrouperLoaderLog();
full_provisioner.startFullSyncOfAllGroupsAndWaitForCompletion(hib3LoaderLog);
```

Full sync of a single group:

Full Sync Provisioner using GrouperMessagingEngine (Faster)

```
provisioner_name="xyz"; // Whatever your provisioner is called in grouper_loader.properties
group_name="this:is:a:group";

gs = GrouperSession.startRootSession();
edu.internet2.middleware.grouperClient.messaging.GrouperMessagingEngine.send(
new edu.internet2.middleware.grouperClient.messaging.GrouperMessageSendParam()
.assignGrouperMessageSystemName(GrouperBuiltinMessagingSystem.BUILTIN_NAME)
.assignQueueType(edu.internet2.middleware.grouperClient.messaging.GrouperMessageQueueType.queue)
.assignQueueOrTopicName("pspng_full_sync_" + provisioner_name +"_asap")
.addMessageBody(group_name));
```

Full Sync Provisioner using FullSyncer (Slower, but also works)

```
provisioner_name="xyz"; // Whatever your provisioner is called in grouper_loader.properties
group_name="this:is:a:group";

gs = GrouperSession.startRootSession();
full_provisioner=edu.internet2.middleware.grouper.pspng.FullSyncProvisionerFactory.getFullSyncer
(provisioner_name);
qt=edu.internet2.middleware.grouper.pspng.FullSyncProvisioner$QUEUE_TYPE;
full_provisioner.scheduleGroupForSync(qt.ASAP, group_name, "optExtReference", "manualFullSync");
//optExtReference can be an issue tracking number or ""
```

Adjusting log level. You can edit the log4j.properties, or you can adjust pspng's logging within the interactive gsh session:

Full Sync Provisioner

```
pspng_logger=org.slf4j.LoggerFactory.getLogger("edu.internet2.middleware.grouper.pspng");
pspng_logger.logger.setLevel(org.apache.log4j.Level.DEBUG)
layout = new org.apache.log4j.PatternLayout("%-5p %c %x - %m%n");
pspng_logger.logger.addAppender(new org.apache.log4j.ConsoleAppender(layout));
```

Provisioning "empty" groupOfNames groups

The PSP supported provisioning "empty" groups with objectClass groupOfNames by using a configuration

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

Doing so would allow the PSP to provision a group with no members in Grouper into LDAP, for example

```
dn: cn=aGroup,ou=aStem,dc=my,dc=org
objectClass: groupOfNames
objectClass: eduMember
cn: aGroup
member:
```

The value for the member attribute in that record is the null DN (also known as the zero-length DN). The null DN is allowed DN syntax.

As of Grouper 2.5.51, the PSP-NG can be made to accomplish the same if the provisioner configuration includes configuration options (properties) like the following (change as appropriate with the name of your provisioner and your use case needs):

```
changeLog.consumer.pspng_groupOfNames.supportsEmptyGroups = true
changeLog.consumer.pspng_groupOfNames.groupCreationLdifTemplate = dn: ${utils.bushyDn(group.name, "cn", "ou")}
||cn: ${group.extension}||objectclass: groupOfNames||objectclass: eduMember||member:||description: ${group.
idIndex}
changeLog.consumer.pspng_groupOfNames.allowEmptyDnAttributeValues = true
changeLog.consumer.pspng_groupOfNames.removeNullDnFromGroupLdifCreationTemplate = true
```

The supportsEmptyGroups property must be true so that the PSP-NG will attempt to create a record in LDAP for a group with no members in Grouper. The groupCreationLdifTemplate includes the attribute member with no value. The allowEmptyDnAttributeValues allows the empty or null DN value to be included in the add group command sent to the LDAP server. The removeNullDnFromGroupLdifCreationTemplate causes the member attribute with null DN to be filtered out of the LDIF when the LDIF is later used as part of group update and synchronization operations.

Provisioner Templates

The syntax for each line is:

`changelog.consumer.<provisioner-name>.<propertyName> = value`

Bushy DNs

In the provisioner snippets below, the groups will be created hierarchically. To get flatter group names, you can use the utils.bushyDn function. For example:

```
changeLog.consumer.<provisioner-name>.groupCreationLdifTemplate = dn: cn=${group.name},${utils.bushyDn(group.
name, "cn", "ou")},dc=example,dc=edu
```

GROUP OF UNIQUE NAMES



This is most likely the provisioner needed for most LDAP servers, such as OpenLDAP, OpenDJ, etc.

```
changeLog.consumer.pspng_groupOfUniqueNames.class = edu.internet2.middleware.grouper.pspng.
PspChangelogConsumerShim
changeLog.consumer.pspng_groupOfUniqueNames.type = edu.internet2.middleware.grouper.pspng.LdapGroupProvisioner
changeLog.consumer.pspng_groupOfUniqueNames.quartzCron = 0 * * * * ?
changeLog.consumer.pspng_groupOfUniqueNames.ldapPoolName = opendj
changeLog.consumer.pspng_groupOfUniqueNames.memberAttributeName = uniqueMember
# changeLog.consumer.pspng_posixGroup.memberAttributeValueFormat = ${ldapUser.getStringValue("uid")}
changeLog.consumer.pspng_groupOfUniqueNames.memberAttributeValueFormat = ${ldapUser.getDn()}
changeLog.consumer.pspng_groupOfUniqueNames.groupSearchBaseDn = ou=grouper,ou=groups,dc=example,dc=edu
changeLog.consumer.pspng_groupOfUniqueNames.allGroupsSearchFilter = objectclass=groupOfUniqueNames
changeLog.consumer.pspng_groupOfUniqueNames.singleGroupSearchFilter = (&(objectclass=groupOfUniqueNames)
(cn=${group.name}))
changeLog.consumer.pspng_groupOfUniqueNames.groupSearchAttributes=cn,gidNumber,objectclass
changeLog.consumer.pspng_groupOfUniqueNames.groupCreationLdifTemplate = dn: cn=${group.name}||cn: ${group.name}
||objectclass: groupOfUniqueNames
changeLog.consumer.pspng_groupOfUniqueNames.userSearchBaseDn = cn=users,dc=example,dc=edu
changeLog.consumer.pspng_groupOfUniqueNames.userSearchFilter = uid=${subject.id}
```

POSIX GROUPS

```
changeLog.consumer.pspng_posixGroup.class = edu.internet2.middleware.grouper.pspng.PspChangelogConsumerShim
changeLog.consumer.pspng_posixGroup.type = edu.internet2.middleware.grouper.pspng.LdapGroupProvisioner
changeLog.consumer.pspng_posixGroup.quartzCron = 0 * * * * ?
changeLog.consumer.pspng_posixGroup.ldapPoolName = opendj
changeLog.consumer.pspng_posixGroup.memberAttributeName = memberUid
changeLog.consumer.pspng_posixGroup.memberAttributeValueFormat = ${ldapUser.getStringValue("uid")}
changeLog.consumer.pspng_posixGroup.groupSearchBaseDn = ou=grouper-posix,ou=groups,dc=example,dc=edu
changeLog.consumer.pspng_posixGroup.allGroupsSearchFilter = objectclass=posixGroup
changeLog.consumer.pspng_posixGroup.singleGroupSearchFilter = (&(objectclass=posixGroup)(cn=${group.name}))
changeLog.consumer.pspng_posixGroup.groupSearchAttributes=cn,gidNumber,objectclass
# Obviously, gidNumber should be based on a grouper-group attribute
changeLog.consumer.psong_posixGroup.groupCreationLdifTemplate = dn: cn=posix-${group.name}||cn: posix-${group.
name}||objectclass: posixGroup||objectclass: groupOfNames||gidNumber: ${group.idIndex}
changeLog.consumer.pspng_posixGroup.userSearchBaseDn = cn=users,dc=example,dc=edu
changeLog.consumer.pspng_posixGroup.userSearchFilter = uid=${subject.id}
```

ACTIVE DIRECTORY GROUPS

```
changeLog.consumer.pspng_activedirectory.class = edu.internet2.middleware.grouper.pspng.PspChangelogConsumerShim
changeLog.consumer.pspng_activedirectory.type = edu.internet2.middleware.grouper.pspng.LdapGroupProvisioner
changeLog.consumer.pspng_activedirectory.quartzCron = 0 * * * * ?
changeLog.consumer.pspng_activedirectory.ldapPoolName = active_directory
changeLog.consumer.pspng_activedirectory.isActiveDirectory = true
changeLog.consumer.pspng_activedirectory.memberAttributeName = member
changeLog.consumer.pspng_activedirectory.memberAttributeValueFormat = ${ldapUser.getDn()}
changeLog.consumer.pspng_activedirectory.groupSearchBaseDn = ou=grouper,ou=groups,dc=example,dc=edu
changeLog.consumer.pspng_activedirectory.allGroupsSearchFilter = objectclass=group
changeLog.consumer.pspng_activedirectory.singleGroupSearchFilter = (&(objectclass=group)(gidNumber=${group.
idIndex}))
changeLog.consumer.pspng_activedirectory.groupCreationLdifTemplate = dn: cn=${group.name}||cn: ${group.
extension}||objectclass: group||gidNumber: ${group.idIndex}||displayname:${group.displayName.replace(group.
parentStemName.toString()+":","")}||description:${org.apache.commons.lang3.StringUtils.abbreviate(group.
description == null ? null : group.description.replaceAll("\\r\\n"," ").replaceAll("\\n"," ").replaceAll
("\\r"," "), 900)}
changeLog.consumer.pspng_activedirectory.userSearchBaseDn = cn=users,dc=example,dc=edu
changeLog.consumer.pspng_activedirectory.userSearchFilter = samAccountName=${subject.id}
```

USER ATTRIBUTES

```
changeLog.consumer.pspng_attributes.class = edu.internet2.middleware.grouper.pspng.PspChangelogConsumerShim
changeLog.consumer.pspng_attributes.type = edu.internet2.middleware.grouper.pspng.LdapAttributeProvisioner
changeLog.consumer.pspng_attributes.quartzCron = 0 * * * * ?
changeLog.consumer.pspng_attributes.retryOnError = true
changeLog.consumer.pspng_attributes.ldapPoolName = opendj
changeLog.consumer.pspng_attributes.provisionedAttributeName = eduPersonEntitlement
changeLog.consumer.pspng_attributes.provisionedAttributeValueFormat = g:${group.name}
changeLog.consumer.pspng_attributes.userSearchBaseDn = cn=users,dc=example,dc=edu
changeLog.consumer.pspng_attributes.userSearchFilter = uid=${subject.id}
changeLog.consumer.pspng_attributes.allProvisionedValuesPrefix=g:
```