# Grouper LDAP provisioner



The info on this page applies to Grouper v4 and above. See Also Grouper Provisioning Framework



The LDAP provisioner in Grouper v4+ replaces the old PSPNG.

The major differences from PSPNG is how groups are identified as provisionable, bulk operations for performance, using sync objects for caching and performance, configuration using the UI, etc.

## High level summary

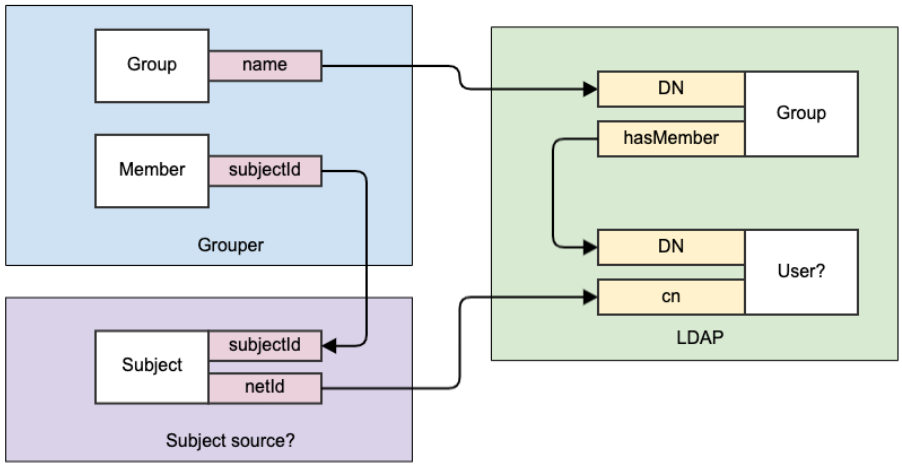
- First configure an external system for the target LDAP.
- Supports both memberships in groups (e.g. using the member attribute) and memberships in entities (e.g. using the memberOf or eduPersonEntitlement attributes). Also referred to as groupAttributes vs entityAttributes.
- Groups can be flat or bushy in LDAP.
- Configure a "name" field in the object model to represent the LDAP object DN for groups and entities.
- All other attributes in the LDAP object are attributes (rather than fields) in the object model.
    - For example, for groups, you may have attributes such as:
        - cn
        - objectClass (based on staticValues)
        - gidNumber (could be configured as the matching id and search attribute)
        - description (optional)
        - member (configured as the membership attribute if groupAttributes)
    - And for entities, you may have attributes such as:
        - uid (could be configured as the matching id and search attribute)
        - memberOf or eduPersonEntitlement (configured as the membership attribute if entityAttributes)
        - If creating entities, then you may have several other attributes such as givenName, sn, objectClass, etc.
- If you need to lookup an object in LDAP to get the DN, use a "group link" or "entity link"
    - Also store the DN in the groupToId2 sync bucket or the memberToId2 sync bucket

## FAQ

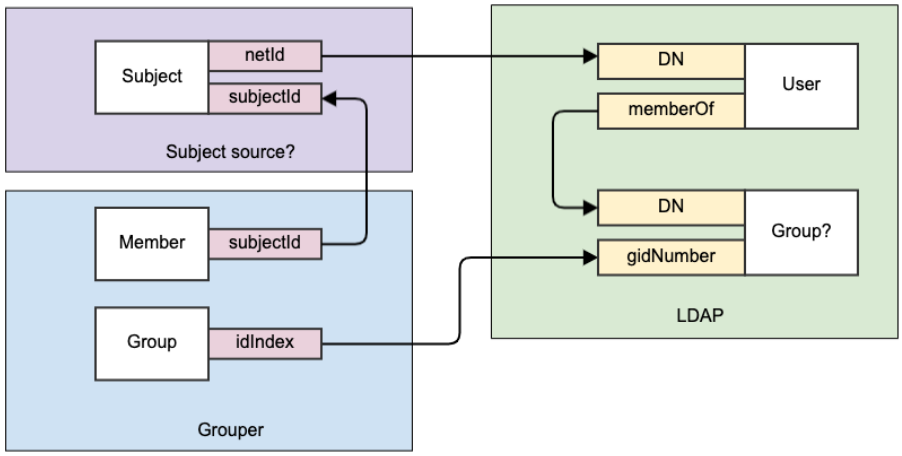
How can I provision a boolean to LDAP

- Use a string with TRUE or FALSE

## LDAP provisioning types




Type: groupAttributes




Type: entityAttributes

## Default Translations

- The "name" field for groups and entities is the DN of the LDAP entry. For groups, the DN is generated by default if the translation type is grouperProvisioningGroupField
  - If an override DN is configured for a group, then that will be used.
  - Otherwise for flat provisioning, it will be in the format: cn=<fieldValue>,baseDN
    - cn is the default RDN attribute for groups but you can change this.
    - The grouperProvisioningGroupField field for the name field is usually expected to be "name" to allow the DN to be generated using the group name. This is the fieldValue.
    - The grouperProvisioningGroupField field for the RDN attribute (usually cn) is usually "name" as well.
    - The baseDN is the Group search base DN.
    - For example the Grouper group app:service1:testGroup might be cn=app:service1:testGroup,ou=groups,dc=example,dc=edu
  - Otherwise for bushy provisioning, it will be in the format: cn=<cnValue>,ou=<parentFolderExtension1>,ou=<parentFolderExtension2>,..., baseDN
    - cn is the default RDN attribute for groups but you can change this.
    - ou is the default RDN attribute for folders but you can change this as well.
    - The grouperProvisioningGroupField field for the name field is usually expected to be "name" to allow the DN to be generated using the group name.
    - The grouperProvisioningGroupField field for the RDN attribute (usually cn) is usually "extension". This determines the cnValue.
    - For example the Grouper group app:service1:testGroup might be cn=testGroup,ou=service1,ou=app1,ou=groups,dc=example, dc=edu
- If you allow DN overrides, the RDN attribute for groups (usually cn) will be automatically computed for groups with an override DN.

## Translation examples

RDN value adjusting group name (remove prefix)

```
${edu.internet2.middleware.grouper.util.GrouperUtil.ldapEscapeRdnValue(edu.internet2.middleware.grouper.util.
GrouperUtil.stripPrefix(grouperProvisioningGroup.name, 'w:e:'))}
```

DN adjusting group name (remove prefix)

```
${edu.internet2.middleware.grouper.util.GrouperUtil.ldapBushyDn(edu.internet2.middleware.grouper.util.
GrouperUtil.stripPrefix(grouperProvisioningGroup.name, 'w:e:'), 'cn', 'ou', true, false) + ',ou=groups,
dc=school,dc=edu'}
```

## LDAP specific configuration

- Ldap provisioning type
  - groupAttributes - group ldap object has attribute to hold memberships (e.g. member)
  - entityAttributes - user ldap object has attribute to hold memberships (e.g. memberOf, eduPersonEntitlement, etc)
- Group search base DN
  - Base DN containing groups
- Group search filter
  - Find a single group. You can use the variable 'targetGroup'. e.g. (&(gidNumber=${targetGroup.retrieveAttributeValue('gidNumber')}) (objectClass=groupOfNames))
  - Note, if the search filter is simply the search attribute, then you can leave this blank.
- Group search all filter
  - Find all groups filter. If you leave this blank it will default to searching for containing the search attribute and if there is an objectclass attribute then those will be included in the filter too. e.g. (&(gidNumber=*)(objectClass=posixGroup)(objectClass=top))
- Allow DN override
  - When marking a group as provisionable, allow the DN to be set to a different value than the default (to point a Grouper group to an arbitrary LDAP group as a one-off).
  - Default value is 'false'.
- RDN attribute for groups
  - The RDN attribute for group objects in LDAP.
  - Defaults to cn.
- Group DN type
  - flat for bushy
- RDN attribute for folders
  - The RDN attribute for folder objects in LDAP.
  - e.g. If the RDN attribute for folders is "ou" and the RDN attribute for groups is "cn", then the DN for the Grouper group app:foo:service:policy:foo_user would be cn=foo_user,ou=policy,ou=service,ou=foo,ou=app,<group base DN>.
  - Only applicable if provisioning type is bushy
  - Defaults to ou.
- Object classes for folders
  - The objectClass values (comma-separated) to add to folder objects in LDAP.
  - Only applicable if provisioning type is bushy
  - Default value is 'top, organizationalUnit'.
- Entity search base DN
  - Base DN containing entites
- Entity search filter
  - Find a single entity. You can use the variable 'targetEntity'. e.g. (&(employeeId=${targetGroup.retrieveAttributeValue('employeeId')}) (objectClass=eduPerson))
  - Note, if the search filter is simply the search attribute, then you can leave this blank.
- Entity search all filter
  - Find all entities filter. If you leave this blank it will default to searching for containing the search attribute and if there is an objectclass attribute then those will be included in the filter too. e.g. (&(employeeId=*)(objectClass=eduPerson)(objectClass=top))

## Default values

Fields and attributes can have default values.  This is particularly important for some LDAP products that always require a member for groups.  In other words, if you have a group in Grouper with no members and that is provisioned to LDAP, the LDAP group must have a member.  You can have a default value for the member attribute that could either be some DN in your LDAP system.  Or if supported by your LDAP, you could use <emptyString> if the default value should be an empty string.

## Best practices

Lets craft some best practices

| Best practice | Description |
| --- | --- |

| | |
|---|---|
| If using AD, use bushy provisioning | CN in AD can only be 64 which the group name frequently is longer than |
| If bushy provisioning, provision the group name somehow | An attribute could hold the group name, or sAMAccountName could hold group name with colons as underscores |
| Should put gidNumber or Grouper UUID in an LDAP attribute | Helps with renaming |
| If you want to track which groups are from which provisioners you could provision the provisionerId to an attribute | Could be used in a filter or so its easy to see which grouper provisioner created/or/manages that group<br>Note: you dont need this if all groups for provisioner are in a dedicated OU |

## Supported LDAP DAO Operations

These are the supported LDAP DAO operations. The operations that are actually called are based on how you've configured the provisioner. For example, you may or may not want Grouper to insert entities to LDAP.

- retrieveAllGroups
    - Used to retrieve all groups from LDAP during a full sync.
    - Query based on the "Group search all filter" configuration. If blank, the query is generated based on the search attribute and object classes defined.
- retrieveGroups
    - Retrieve a list of groups from LDAP in batches of 100 (by default).
    - Query based on the "Group search filter" configuration. If blank, the query is generated based on the search attribute.
- insertGroup
    - Add a group to LDAP
    - The DN is based on the "name" field in the targetGroup.
    - Attributes in the targetGroup are set as attributes in LDAP. This includes an attribute you may be using to store memberships in the group (e.g. member). But also other attributes such as objectClass, description, gidNumber, etc.
    - For bushy provisioning, this will create parent OUs.
- deleteGroup
    - Delete a group in LDAP
    - For bushy provisioning, this will delete empty parent OUs.
- updateGroup
    - Update a group in LDAP. For example, update memberships or other attributes.
    - Supports DN renames.
    - Updates are made in LDAP in batches (default batch size is 100). So if you're adding 1000 members to a group, that would be 10 updates.
    - If there's a failure in a batch, then the updates are done individually.
- retrieveAllEntities
    - Used to retrieve all entities from LDAP during a full sync.
    - Query based on the "Entity search all filter" configuration. If blank, the query is generated based on the search attributes and object classes defined.
- retrieveEntities
    - Retrieve a list of entities from LDAP in batches of 100 (by default).
    - Query based on the "Entity search filter" configuration. If blank, the query is generated based on the search attribute.
- insertEntity
    - Add an entity to LDAP
    - The DN is based on the "name" field in the targetEntity.
    - Attributes in the targetEntity are set as attributes in LDAP. This includes an attribute you may be using to store memberships in the entity (e.g. eduPersonEntitlement or memberOf). But also other attributes such as objectClass, givenName, sn, etc.
- deleteEntity
    - Delete an entity in LDAP
- updateEntity
    - Update an entity in LDAP. For example, update memberships or other attributes.
    - Supports DN renames.
    - Updates are made in LDAP in batches (default batch size is 100).
    - If there's a failure in a batch, then the updates are done individually.
- retrieveMembership
    - Retrieve a single membership from LDAP (based on either a group or entity).
    - The base DN of the query is the DN of the group or entity. And the search checks to see if the membership is in the object. e.g. (member=something) or (eduPersonEntitlement=something)
- retrieveMembershipsByGroups
    - Just calls retrieveGroups

## Low level logging

If you are troubleshooting issues, you can enable low level Ldaptive logging in the Advanced configuration to get details about the actual operations sent to LDAP and the responses.  The logs will go to the container logs.

- Log target commands always
  - Log the low level commands to the target for all commands (successes and failures). This should only be enabled while troubleshooting.
  - Default value is 'false'.
- Log target commands on error
  - Log the low level commands to the target for errors. This should only be enabled while troubleshooting. This has performance implications even for non error transactions.
  - Default value is 'false'.

Example:

```
2021-11-04 15:28:40,182: [Thread-21] INFO  GrouperProvisioningLogCommands.infoLog(25) -  - Command log for
provisioner 'ldapProvTest' - 'u5vmmuhb', retrieveAllEntities: Ldaptive searchRequest: [org.ldaptive.
SearchRequest@-349354149::baseDn=ou=People,dc=example,dc=edu, searchFilter=[org.ldaptive.
SearchFilter@1043503778::filter=(&(objectClass=person)(uid=*)), parameters={}], returnAttributes=[cn,
objectClass, sn, uid], searchScope=SUBTREE, timeLimit=0, sizeLimit=0, derefAliases=null, typesOnly=false,
binaryAttributes=null, sortBehavior=UNORDERED, searchEntryHandlers=null, searchReferenceHandlers=null,
controls=null, referralHandler=null, intermediateResponseHandlers=null]
```

## GSH troubleshooting

Run an ldap filter

```
    import edu.internet2.middleware.grouper.ldap.*;
    GrouperSession grouperSession = GrouperSession.startRootSession();
    List<LdapEntry> ldapEntries = LdapSessionUtils.ldapSession().list("personLdap", "ou=People,dc=example,
dc=edu", LdapSearchScope.SUBTREE_SCOPE, "uid=*", (String[])(Object)GrouperUtil.toArray("uid"), null);
    System.out.println(GrouperUtil.length(ldapEntries));
```

## Migrate from pspng

Run this:

```
./gsh.sh -pspngAttributesToProvisioningAttributes pspngConfigId provisioningFrameworkConfigId
readonly|notReadonly deleteOrphans|dontDeleteOrphans
```

deleteOrphans means remove provisioning framework provisionable assignments that do not exist in pspng.  dontDeleteOrphans means leave those alone and the provisioning framework provisionable assignments are potentially a superset of pspng

Once the new provisioner is provisioning the data, turn off the pspng jobs

## Caching

Sync objects can cache information in LDAP.  The cached data below are examples.  You can cache whatever data you want.  Synced from full sync (if doesnt exist or if errors), incremental (if doesnt exist or if errors), and the nightly (scheduled) subject resolution daemon (full refresh)

| Object | Field | Cached data |
|---|---|---|
| gcGrouperSyncGroup | groupToId2 | group DN |
| gcGrouperSyncGroup | groupToId3 | whatever attribute value the user attribute refers to |
| gcGrouperSyncGroup | groupFromId2 | ldap group object attribute value that looks up group |
| gcGrouperSyncMember | memberToId2 | user DN |
| gcGrouperSyncMember | memberToId3 | whatever attribute value the group attribute refers to users as |

| | | |
|---|---|---|
| gcGrouperSyncMember | memberFromId2 | ldap person object attribute value that looks up user |
| gcGrouperSyncMember | memberFromId3 | subject attribute value that helps look up user |

# Developer Documentation Below (old documentation)

## Configuration

Common config attributes for LDAP are below.

| Config | Example | Description | Notes |
|---|---|---|---|
| class | edu.internet2.middleware.grouper.app.ldapProvisioning.LdapSync | Class extends the base provisioner class | This class informs configuration decisions. Required. Read-only. |
| hasSubjectLink | true<br><br>false | If the subject API is needed to resolve attribute on subject | required, drives requirements of other configurations. defaults to false. |
| hasTargetUserLink | true<br><br>false | If subjects need to be resolved in the target before provisioning | defaults to false. required.<br><br>show if groupMemberships |
| hasTargetGroupLink | true<br><br>false | If groups need to be resolved in the target before provisioning | defaults to false. required.<br><br>show if userAttributes |
| subjectSourcesToProvision | pennperson | subject sources to provision | required. defaults to all except g:gsa, grouperExternal, g:isa, localEntities. comma separated list. checkboxes. |
| createMissingUsers | true or false | | defaults false. required.<br><br>show if userAttributes<br>or hasTargetUserLink |
| createMissingGroups | true or false | | defaults to true.  required.<br><br>show if groupMemberships or hasTargetGroupLink |
| deleteInTargetIfInTargetAndNotGrouper | true or false | if groups in full sync should be deleted if in group all filter and not in grouper<br>or for attributes delete other attribute not provisioned by grouper | default to false |
| deleteInTargetIfDeletedInGrouper | true or false | if groups that were created in grouper were deleted should it be deleted in ldap?<br>or for attributes, delete attribute value if deleted in grouper | default to true |
| membershipFields | members<br><br>read,admin<br><br>update,admin<br><br>admin | if provisioning normal memberships or privileges | default to "members" for normal memberships |
| userSearchFilter | ldap example:<br><br>(&(objectClass=person)(uid=${targetEntity.retrieveAttributeValue('uid')})) | how to find a user | optional. show if userAttributes or hasTargetUserLink |
| userSearchAllFilter | ldap example:<br><br>(&(objectClass=person)(uid=*)) | filter users when searching all | optional. show if userAttributes or hasTargetUserLink |
| groupSearchFilter | ldap example:<br><br>(&(objectClass=group)(gidNumber=${targetGroup.retrieveAttributeValue('gidNumber')})) | find a single group (other than by DN) | optional. show if groupMemberships or hasTargetGroupLink |
| groupSearchAllFilter | ldap example:<br><br>(&(objectclass=group)(gidNumber=*)) | find all groups | optional. show if groupMemberships or hasTargetGroupLink |
| refreshGroupLinkIfLessThanAmount | 20 | refresh target group link if less than this amount | show if hasTargetGroupLink |
| common.groupLink.groupFromId2 | ${targetGroup.getName()} | Target group link - groupFromId2 | show if hasTargetGroupLink |

| | | | |
|---|---|---|---|
| common.groupLink.groupFromId3 | | Target group link - groupFromId3 | show if hasTargetGroupLink |
| common.groupLink.groupToId2 | | Target group link - groupToId2 | show if hasTargetGroupLink |
| common.groupLink.groupToId3 | | Target group link - groupToId3 | show if hasTargetGroupLink |
| refreshEntityLinkIfLessThanAmount | 20 | refresh target user link if less than this amount | show if hasTargetUserLink |
| common.entityLink.memberFromId2 | ${targetEntity.getName()} | Target user link - memberFromId2 | optional<br>show if hasTargetUserLink |
| common.entityLink.memberFromId3 | | Target user link - memberFromId3 | optional<br>show if hasTargetUserLink |
| common.entityLink.memberToId2 | | Target user link - memberToId2 | optional<br>show if hasTargetUserLink |
| common.entityLink.memberToId3 | | Target user link - memberToId3 | optional<br>show if hasTargetUserLink |
| refreshSubjectLinkIfLessThanAmount | 20 | refresh subject link if less than this amount | show if hasSubjectLink |
| common.subjectLink.<br>memberFromId2 | | Subject link - memberFromId2 | optional<br>show if hasSubjectLink |
| common.subjectLink.<br>memberFromId3 | | Subject link - memberFromId3 | optional<br>show if hasSubjectLink |
| common.subjectLink.memberToId2 | | Subject link - memberToId2 | optional<br>show if hasSubjectLink |
| common.subjectLink.memberToId3 | | Subject link - memberToId3 | optional<br>show if hasSubjectLink |
| groupAllowedToAssign | (group name) | group allowed to assign<br>only grouper system admin is allowed when no specific group is allowed to assign the given target | optional |
| allowAssignmentsOnlyOnOneStem | true or false | allow assignment only on one stem | default to false |
| readOnly | true or false | read only | default to false |
| debugLog | true or false | enable debug log | default to false |
| logAllObjectsVerbose | true or false | log all objects verbose | default to false |
| targetEntityAttributeCount | 5 | number of attributes for users | default to 0<br>Should be 0-20<br>show if userAttributes or hasTargetUser Link |
| targetGroupAttributeCount | 5 | number of attributes for groups | default to 0<br>Should be 0-20<br>show if groupMemberships or hasTarge tGroupLink |
| targetEntityAttribute.[0-19].name | uid | name of the attribute | required |
| targetEntityAttribute.[0-19].<br>isFieldElseAttribute | true or false | is field else attribute? | default to false |
| targetEntityAttribute.[0-19].valueType | string | value type | required<br>Should be string or int or long |
| targetEntityAttribute.[0-19].insert | true or false | insert attribute? | default to false |
| targetEntityAttribute.[0-19].update | true or false | update attribute? | default to false |
| targetEntityAttribute.[0-19].delete | true or false | delete attribute? | default to false |
| targetEntityAttribute.[0-19].select | true or false | select attribute? | default to false |
| targetEntityAttribute.[0-19].<br>matchingId | true or false | matching id attribute? | default to false |

| | | | |
|---|---|---|---|
| targetEntityAttribute.[0-19].<br>multiValued | true or false | multi-valued attribute? | default to false |
| targetEntityAttribute.[0-19].<br>membershipAttribute | true or false | is this the membership attribute? | default to false |
| targetEntityAttribute.[0-19].<br>translateExpressionFromMembership | ${gcGrouperSyncGroup.<br>getGroupToId2()} | translate expression from membership | show and require if<br>membershipAttribute |
| targetEntityAttribute.[0-19].<br>translateExpressionCreateOnly | ${grouperUtil.toSet('top', 'person')} | translate expression when creating objects only | show if not membershipAttribute and insert = true |
| targetEntityAttribute.[0-19].<br>translateExpression | ${grouperProvisioningEntity.<br>getSubjectId()} | translate expression otherwise | show and require if not membershipAttribute |
| targetGroupAttribute.[0-19].name | gidNumber | name of the attribute | required |
| targetGroupAttribute.[0-19].<br>isFieldElseAttribute | true or false | is field else attribute? | default to false |
| targetGroupAttribute.[0-19].<br>valueType | long | value type | required<br><br>Should be string or int or long |
| targetGroupAttribute.[0-19].insert | true or false | insert attribute? | default to false |
| targetGroupAttribute.[0-19].update | true or false | update attribute? | default to false |
| targetGroupAttribute.[0-19].delete | true or false | delete attribute? | default to false |
| targetGroupAttribute.[0-19].select | true or false | select attribute? | default to false |
| targetGroupAttribute.[0-19].<br>matchingId | true or false | matching id attribute? | default to false |
| targetGroupAttribute.[0-19].<br>multiValued | true or false | multi-valued attribute? | default to false |
| targetGroupAttribute.[0-19].<br>membershipAttribute | true or false | is this the membership attribute? | default to false |
| targetGroupAttribute.[0-19].<br>translateExpressionFromMembership | ${gcGrouperSyncMember.<br>getMemberToId2()} | translate expression from membership | show and require<br>if membershipAttribute |
| targetGroupAttribute.[0-19].<br>translateExpressionCreateOnly | ${grouperUtil.toSet('top',<br>'posixGroup')} | translate expression when creating objects only | show if not membershipAttribute and insert = true |
| targetGroupAttribute.[0-19].<br>translateExpression | ${grouperProvisioningGroup.<br>getIdIndex()} | translate expression otherwise | show and require if not membershipAttribute |

## Discussion topics

- Should we use DAO for LDAP so we can unit test everything but the LDAP stuff? Also for dry run? 🙂
  Yes

- User cache / group cache / subject cache
  - Need a provisioner col for last retrieved
    Update cache with nightly full sync, on error, refresh subject and ldap user from target, and try again
  - How daemon works
    syncing and the cache updating. Note: it can use the subject source data from the nonblocking state inside the blocking state
    1. Nonblocking, unless over a threshold. Few changes, start blocking after know changes, get the state from both sides for those changes, make the changes, unblock.
    2. If over a certain threshold (need algorithm for number of objects: groups or users), block, and recalculate and sync
  - How cached fields stored in db? JSON? cols?
    Keep a col for the lookups to query on, and a col for json representation of group and user
  - Store subject attributes? use member table? add a couple cols? identifier1, identifier2..5? update periodically?
    Lookup attributes will be in the sync tables
- Pulling data from grouper, only certain columns?
  Dont pull from hibernate, just use simple query like grouper_memberships_lw_v but without distinct (maybe groupId and memberId)
- User sync?
  Yes, we should plan for this
- Folder sync
  Do individual groups if its inside a threshold or full refresh from outside the threshold. If group deletes in ldap are needed, a full sync is needed
- Selecting subjects
  - Use the subject cache, and tune that as needed to fully prime
  - If errors resolve subjects without cache
- Batch size, should be at ldap or provisioner level? do we need different for users and groups? e.g. userSearch_batchSize
  Default batch size at LDAP level, override for provisioner. If over threshold, get all, otherwise, cycle through. Could leave "all" filter blank if known not to provision a lot of groups
  - Need a batch size for updates? How do batches work?
    Theres no batch size for objects, one at a time. Attributes have batch size with default at ldap level
  - Can this be in the LDAP framework (and it will batch things up, thats what we do for jdbc) yes

- Supports empty groups, at ldap or provisioner level?
  Could have null member, could delete group if no members.  Default at ldap level.
- Should errors be stored in the sync objects so they can be absorbed?  yes
  - Message for processing errors?  yes and timestamp
  - Group, user, membership errors?  yes, list the error and skip (based on configuration)
- Search result paging is at ldap or provisioner level?  defaults at ldap level
- Do we need an isActiveDirectory flag?  keep this, TODO add back in
- Support dry run?
  For grouper admins to test what will happen if configuration changes.
  For a single group, do a "pretend is provisionable" in ui jvm...
  For single group or user, this would just happen synchronously in ui, or have the ajax updating screen thing (upcoming)
- Assume gid will not change?  name?  dn?  How do we track groups that move?  i.e. in the grouper_sync_group table?
  The cached DN needs to not change until the provisioner can make sure theres a way to reference the old object to do updates and not delete /insert
- Ldif templates, lets change the config of this.  maybe json?  or at least user $newline$?  Need for jexl things?  Escaping?
  Yes, do this, and use $newline$
- Replacing grouperIsAuthoritative to be more explicit
  Yes do this
- Logging like change log consumer or database provisioning
  Yes
- Does memberOf and hasMember always use the DN of the user or group?  generally yes
- Do we need to support updating group and user or will updating the group handle it?  this is an attribute provisioner
- Convert to some simple object structure from ldap?  or use the ldap objects for jexl?  do we need these with the sync groups? We have a low-memory object structure
  - Single valued attributes strings unless configured attributes are multivalued (use arrays since lightweight) Yes, single valued objects or array of strings
  - Last updated columns in ldap?  numbers: number of millis from epoch  Not used
  - User
    - Bean with single and multivalued attribute Similar to pspng, bean with dn and name/value pair attributes
- Can we support getting memberships or privileges (e.g. ADMIN/READ) Yes, interface to get memberships
- Option to constrain to subject source? Yes
- Do we know the types of ldap attributes?  Is there a metadata schema query? Shilen says generally strings
- Group DNs, can it be dynamic?  Can it be handled upstream in provisioning framework? We would like it to happen in the provisioning framework.  Note, complicates the all group filter
- Is memberOf or hasMember the only multi-valued attributes?  do types of attributes matter?  e.g. integer vs string?  objectclass is multivalued, and memberships, other attributes are arrays of strings

## Translation example

Make a bushy DN and take off a prefix

```
${edu.internet2.middleware.grouper.util.GrouperUtil.ldapBushyDn(edu.internet2.middleware.grouper.util.
GrouperUtil.stripPrefix(grouperProvisioningGroup.name, 'org:something:'), 'cn', 'ou', true, false) +  ',
OU=Grouper,DC=else,DC=something,DC=net'}
```