Grouper provisioning translations

The info on this page applies to Grouper 2.6 and above.

The comparisons and logic happen on "target objects" which are translated from grouper

Examples

- This will reverse a name, replace the separators (colons) with something else (in this case dot), and maxlength of 64 will just truncate there. This will generate a generally unique name

```
${edu.internet2.middleware.grouper.util.GrouperUtil.stringFormatNameReverseReplaceTruncate
(grouperProvisioningGroup.name, ".", 64)}
```

Roles

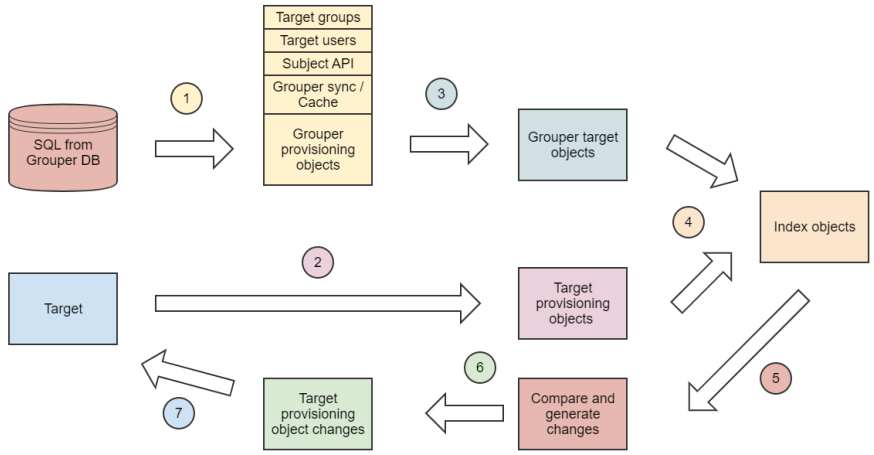
If you need a group membership or privilege on a user in a translation you can use this (4.11.0+, 5.9.0+). These return a boolean

```
${provisioningEntityWrapper.isInGroup('test2:testGroup3')}
${provisioningEntityWrapper.isHasPrivilege('test2:testGroup3', 'admins')}
```

If you need list of users from a group based on membership or privilege, can get a set of strings based on: subjectId, subjectIdentifier0, subjectIdentifier1, subjectIdentifier2, email

```
${provisioningGroupWrapper.groupMembers('test2:testGroup3', 'subjectId')}
${provisioningGroupWrapper.groupPrivilegeHolders('test2:testGroup3', 'updaters', 'subjectIdentifier0')}
```

High level translation

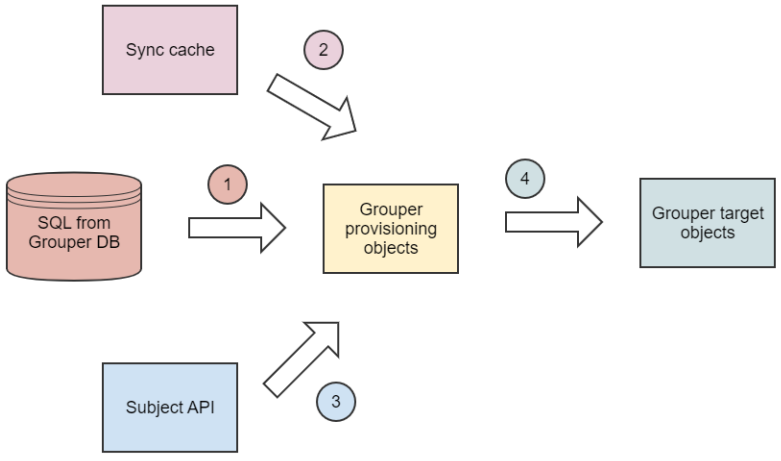


1. Data is retrieved from Grouper, and translated to a standard format
    a. Note: see below, some data might be needed from cache (in DB)
    b. If data is not in cache or there are errors (since cache out of date), lookup in Subject API or Target for users or groups
    c. Abbreviated data is retrieved for Groups/Members/Memberships, but the full sync objects are retrieved
2. Data is retrieved from the Target using the DAO and translated to a "standard" format for that target type (e.g. ldap attributes are put in attributes in the group or entity)
3. Configuration for the grouper provisioning objects translated to grouper target objects. The target object format is specific to the target and documented in the provisioner
    a. If memberships are translated to group attributes or subject attributes, then there is still a reference from the attribute to the membership so things can be tracked
4. Index the target provisioning objects and the grouper target objects so they can be compared
5. The grouper target objects and target provisioning objects are compared by the framework in a generic way
6. If the target and grouper are not in sync, the changes are collected in the form of target provisioning objects
    a. There is a list of individual changes needed in the target (e.g. add this attribute value to this object)
7. The DAO will call CRUD operations (could be batched) to get the target in sync with Grouper

Subject link

"Subject link" is when the subject source needs to be consulted since data to provision is not just the subject id (need attributes cached in grouper or need more up to date data)

Here is the design for subject link from grouper



1. Data is read from Grouper which includes the "sync" tables
2. For entities, this could have some data cached for a subject

```
provisioner.sqlProvTest.syncMemberFromId3AttributeValueFormat = ${subject.attributes['myLdapId']}
```

3. If the data is not cached (or if there is an error), look this up in the subject API (will batch multiple subjects at once)
4. Then once this data is in the "grouper provisioning objects", continue with the translation to the grouper target objects

The provisioning framework will see if the "memberFromId3" is null when getting data from sync cache, those subjects will be resolved and that data will be populated and used

This will translate that to the entity id

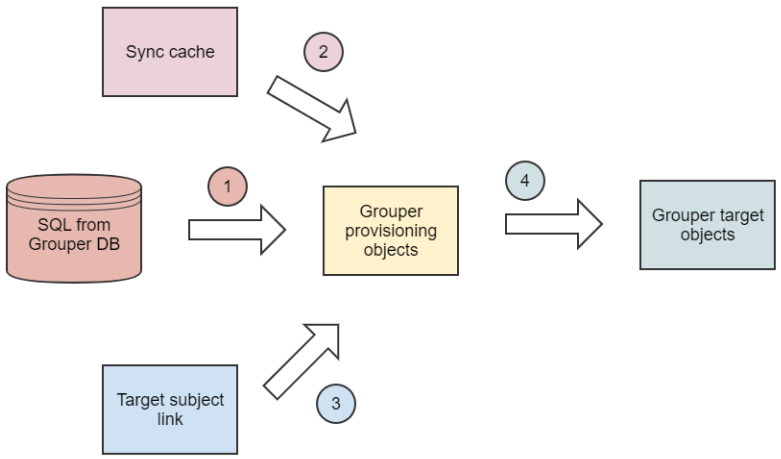
```
#translate from group auto translated to the target format
provisioner.sqlProvTest.grouperToTargetTranslation.0.script = ${grouperTargetEntity.setId(grouperSyncMember.
getMemberFromId3())}

# could be group, membership, or entity
provisioner.sqlProvTest.grouperToTargetTranslation.0.for = entity
```

Target user link

"Target user link" is when the target needs to be consulted since data to provision is not just the subject id or subject api data (need cached in grouper or need more up to date data)

Here is the design for target user link from grouper



1. Data is read from Grouper which includes the "sync" tables
2. For entities, this could have some data cached for a subject

```
# main identifier of the user on the target side
provisioner.sqlProvTest.syncMemberToId2AttributeValueFormat = ${targetEntity.attributes['dn']}

# identifier of the user as referred to by the group
provisioner.sqlProvTest.syncMemberToId3AttributeValueFormat = ${targetEntity.attributes['uid']}
```

3. If the data is not cached (or if there is an error), look this up in the target (will batch multiple entities at once)
4. Then once this data is in the "grouper provisioning objects", continue with the translation to the grouper target objects

The provisioning framework will see if the "memberToId2" or "memberToId3" is null when getting data from sync cache, those entities will be resolved and that data will be populated and used

This will translate that to the entity id

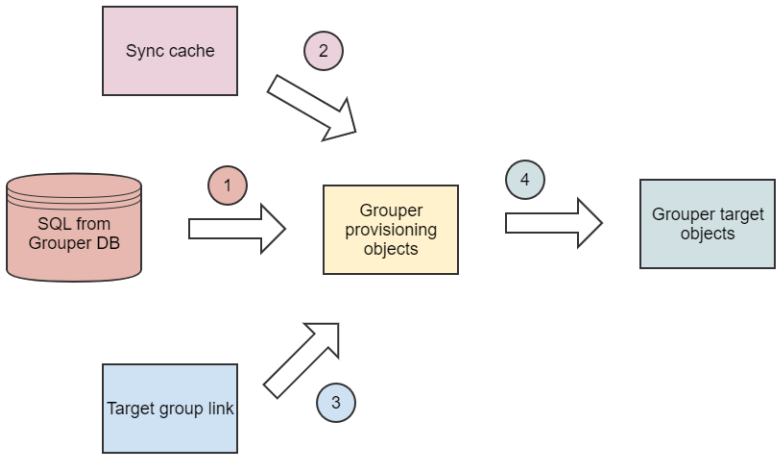
```
#translate from group auto translated to the target format
provisioner.sqlProvTest.grouperToTargetTranslation.0.script = ${grouperTargetEntity.setId(grouperSyncMember.
getMemberToId3())}

# could be group, membership, or entity
provisioner.sqlProvTest.grouperToTargetTranslation.0.for = entity
```

Target group link

"Target group link" is when the target needs to be consulted since data to provision is not just the group uuid, name, id index, etc data (need cached in grouper or need more up to date data)

Here is the design for target group link from grouper



1. Data is read from Grouper which includes the "sync" tables
2. For groups, this could have some data cached from the target

```
# main identifier of the user on the target side
provisioner.sqlProvTest.syncGroupToId2AttributeValueFormat = ${targetGroup.attributes['dn']}
```

3. If the data is not cached (or if there is an error), look this up in the target (will batch multiple entities at once)
4. Then once this data is in the "grouper provisioning objects", continue with the translation to the grouper target objects

The provisioning framework will see if the "groupToId2" or "groupToId3" is null when getting data from sync cache, those entities will be resolved and that data will be populated and used

This will translate that to the group id

```
#translate from group auto translated to the target format
provisioner.sqlProvTest.grouperToTargetTranslation.0.script = ${grouperTargetGroup.setId(grouperSyncGroup.
getGroupToId2())}

# could be group, membership, or entity
provisioner.sqlProvTest.grouperToTargetTranslation.0.for = group
```

Beans

ProvisioningGroup

Field	Type	Description

id	String	every group must have an id, what it is known by in the target. if there is an opaque id, then that is best. Otherwise, whatever it is (uuid, name, extension, idIndex, etc). not: this can be manipulated to turn a name into a DN for example
name	String	This is the group name which can be different than the id. If there is no name different from id, this can be blank. this is used for renames
idIndex	Long (integer)	int assigned from grouper to enable renames of target groups
displayName	String	display name for group
attributes	Map<String, TargetAttribute>	list of attributes where there is a string name and a value (object) value can be a scalar or a collection

ProvisioningEntity (e.g. user)

Field	Type	Description
id	String	every entity must have an id, what it is known by in the target
loginId	String	This is the entity loginid which can be different than the id. If there is no loginid different from id, this can be blank. this is used for changing the netId in the target for an existing user. e.g. netId, eppn, email, etc. optional
name	String	name field can be "first last" or whatever the name should be in the target. optional
email	String	email address used for contact to the user. optional
attributes	Map<String, TargetAttribute>	list of attributes where there is a string name and a value (object) value can be a scalar or a collection

ProvisioningMembership

Field	Type	Description
id	String	membership can have id or else the id will be tuple of group id and subject id (optional)
groupId	String	provisioning group id for membership
group	ProvisioningGroup	reference to the group from this membership
entityId	String	provisioning entity id for memberships
entity	ProvisioningEntity	reference to the entity from this membership
attributes	Map<String, TargetAttribute>	list of attributes where there is a string name and a value (object) value can be a scalar or a collection

ProvisioningAttribute

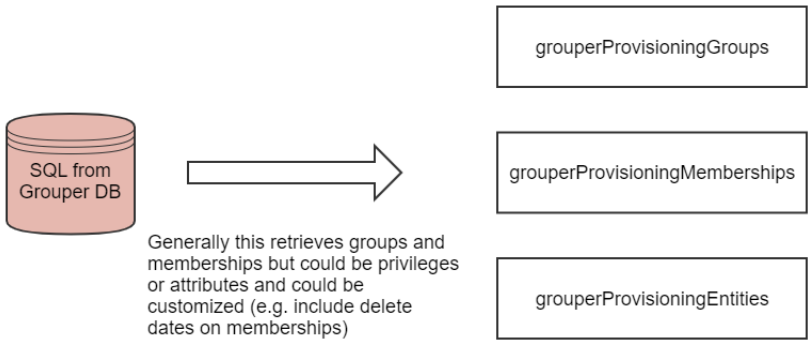
Field	Type	Description
name	String	name of attribute
value	Object (any type, could be multivalued)	value of attribute

Grouper auto translated from SQL

Grouper native comes from SQL queries. This is fairly standard and minorly customizable

Variable (e.g. in EL)	Type	Description

grouperProvisionGroup	ProvisioningGroup	• id: uuid of group • name: group system name • idIndex: group ID index • displayName: display name of group • attribute("description"): description of group
grouperProvisionMembership	ProvisioningMembership	• references to group and entity by group id and member id
grouperProvisionEntity	ProvisioningEntity	• id: memberId • name: name • attribute("description"): subject description • attribute("subjectId"): subject id • attribute("subjectIdentifier0"): subject identifier0



Grouper translated to target

The Grouper provisioning format is processed and the "target object" format needs to be reached.

Variable	Type
grouperTargetGroup	ProvisioningGroup
grouperTargetMembership	ProvisioningMembership
grouperTargetEntity	ProvisioningEntity

First the grouper provisioning groups are cycled through, and for each group, a grouper target group will be created, and a translation will assign fields of the target object from the grouper objects

If a grouperToTargetTranslation "for" is "group" then that translation will be evaluated while looping through groups.

In this example, use the group "name" as the matching ID of the group

```
#translate from group auto translated to the target format
provisioner.sqlProvTest.grouperToTargetTranslation.0.script = ${grouperTargetGroup.setId(grouperProvisionGroup.
getName())}

# could be group, membership, or entity
provisioner.sqlProvTest.grouperToTargetTranslation.0.for = group

#translate from group auto translated to the target format
provisioner.sqlProvTest.grouperToTargetTranslation.1.script = ${grouperTargetGroup.setAttribute("desc",
grouperProvisionGroup.getAttribute("description"))}

# could be group, membership, or entity
provisioner.sqlProvTest.grouperToTargetTranslation.1.for = group
```

If a grouperToTargetTranslation "for" is "entity" then that translation will be evaluated while looping through entities

In this example use the subject id as the matching ID

```
#translate from group auto translated to the target format
provisioner.sqlProvTest.grouperToTargetTranslation.1.script = ${grouperTargetEntity.setId
(grouperProvisionEntity.getAttribute("subjectId"))}

# could be group, membership, or entity
provisioner.sqlProvTest.grouperToTargetTranslation.1.for = entity
```

If a grouperToTargetTranslation "for" is "membership" then that translation will be evaluated while looping through memberships

Target provisioning objects

Depending on the target, fields will be populated and documented for that provisioner. Note: any state needed to make changes in the target need to be stored in these beans

Variable	Type
targetProvisionGroup	ProvisioningGroup
targetProvisionMembership	ProvisioningMembership
targetProvisionEntity	ProvisioningEntity

Target native objects (if exist)

Variable	Type	Notes
targetNativeGroup	?	native target group
targetNativeMembership	?	
targetNativeEntity	?	

Index objects

Use the targetGroup, targetEntity, and targetMembership variable to refer to the grouperTargetGroups, grouperTargetEntity's, grouperTargetMemberships, targetProvisioningGroups, targetProvisioningEntity's, and targetProvisioningMemberships

If you are supporting renames, then the id of the targets need to something opaque and unchanging (e.g. grouper group idIndex or uuid).

Note: you only need to specify the id's for objects that are in the target (e.g. you might not need all three unless there are groups, entities, and memberships in the target).

If something is specified then the group id and entity id will be used for targetGroups and targetEntities. For targetMemberships, it will by default use MultiKey of provisioningGroupId and provisioningEntityId.

The ID cant be null unless its an insert and the target expects a null for insert

The type of the id should be a string, numeric, or a multikey of a few strings / numerics. The ${ } is optional

```
provisioner.sqlProvTest.targetGroupIdExpression = ${targetGroup.retrieveAttributeValueString("groupName")}

provisioner.sqlProvTest.targetMembershipIdExpression = new('edu.internet2.middleware.grouperClient.collections.
MultiKey', targetMembership.retrieveAttributeValueString('group_name'), targetMembership.
retrieveAttributeValueString('subject_id'))

provisioner.sqlProvTest.targetEntityIdExpression = ${targetEntity.retrieveAttributeValueString("subjectId")}
```