Change log consumers

The Grouper change log consists of three tables. The change log temp table is where every Grouper process writes events. The change log temp to change log processes these events, gives them a sequential numeric id, calculates point in time calculations, and moves the data to the change log table. Change log consumers read from the change log table and keep a pointer to their progress in the change log consumer table.

Troubleshooting

Occasionally the changelog consumer may get stuck processing. This can occur when a group has been deleted and a grouperRule cannot find the group to process.

In the UI, these will show up as errors in daemon page:

400 logs found for job name: CHANGE_LOG_consumer_grouperRules

Status	Loaded group	Job type	Start time	End time
Error	N/A	overall	2020-02-12 02:49:06.0	2020-02-12 02:49:06.0
Error	N/A	overall	2020-02-12 02:48:06.0	2020-02-12 02:48:08.0
Error	N/A	overall	2020-02-12 02:47:06.0	2020-02-12 02:47:06.0
Error	N/A	overall	2020-02-12 02:46:06.0	2020-02-12 02:46:06.0
Error	N/A	overall	2020-02-12 02:45:31.0	2020-02-12 02:45:32.0
Error	N/A	overall	2020-02-12 02:44:50.0	2020-02-12 02:44:52.0

Here you can see that every minute, the CHANGE_LOG_consumer_grouperRules has had an Error.

To advance the changelog consumer, view the error message which indicates which changelog entry it was stuck on (scroll all the way to the right in the Grouper UI):

Error: Error processing record 256133454, sequenceNumber: 256133454, java.lang.RuntimeException: …

You can see which change log entry your consumer is stuck at in the grouper_change_log_consumer table (substitute name of change log consumer or select all from table)

```
select LAST_SEQUENCE_PROCESSED from GROUPER_CHANGE_LOG_CONSUMER WHERE NAME = 'grouperRules';
```

Look in the changelog for entries pertaining to the sequence number the consumer got stuck at.

SELECT * FROM GROUPER_CHANGE_LOG_ENTRY WHERE SEQUENCE_NUMBER >= 256133454;

Once you've found the group of entries you want to skip over, update the changelog consumer table for the specific changelog consumer you want to manually advance:

UPDATE GROUPER_CHANGE_LOG_CONSUMER **SET** LAST_SEQUENCE_PROCESSED **=** 256133489 **WHERE** NAME **=** 'grouperRules';

Then return to the grouper miscellaneous status page to monitor the changelog consumer for progress

Change log

This topic is discussed in the Advanced Topics training video.

Grouper can integrate with or provision data to external systems in real-time. This is done using Grouper Notifications, which are based on the Grouper Change Log. Approaches include:

- Use the Provisioning Service Provider (PSP). The PSP is able to incrementally provision one or more target LDAP directories (including Active Directory) based on the Change log. It can also provision SPML targets.
- Use the Grouper ESB Connector or XMPP to implement notifications from the Change log.
- Implement your own custom change log consumer in Java. This provides additional flexibility since you can get direct access to all the information for each change, but does require custom code. This is most useful if you are provisioning a target that cannot be provisioned by the other methods.

Change Log Events

As of Grouper 2.0, the following change log events are supported. Note that Grouper 2.1 no longer has notifications on flattened permissions due to performance concerns. Instead, whenever anything related to a permission changes (including memberships and all the hierarchies that could be involved in forming a permission), change log events are added for all the roles involved. The action name for the change log entry is permissionChangeOnRole.

This diagram shows the Change Log and Notifications in the Grouper Architecture.

Here are some change log events, you should look in source for your version of Grouper to get the full list. See ChangeLogTypeBuiltin.java or the GROUPER_CHANGE_LOG_TYPE table.

Change Log Category	Action Name
attributeAssign	addAttributeAssign
attributeAssign	deleteAttributeAssign
attributeAssignAction	addAttributeAssignAction
attributeAssignAction	deleteAttributeAssignAction
attributeAssignAction	updateAttributeAssignAction
attributeAssignActionSet	addAttributeAssignActionSet
attributeAssignActionSet	deleteAttributeAssignActionSet
attributeAssignValue	addAttributeAssignValue
attributeAssignValue	deleteAttributeAssignValue
attributeDef	addAttributeDef
attributeDef	deleteAttributeDef
attributeDef	updateAttributeDef
attributeDefName	addAttributeDefName
attributeDefName	deleteAttributeDefName
attributeDefName	updateAttributeDefName
attributeDefNameSet	addAttributeDefNameSet
attributeDefNameSet	deleteAttributeDefNameSet
group	addGroup
group	deleteGroup
group	updateGroup
groupField	addGroupField
groupField	deleteGroupField
groupField	updateGroupField
groupType	addGroupType
groupType	deleteGroupType
groupType	updateGroupType

groupTypeAssignment	assignGroupType
groupTypeAssignment	unassignGroupType
member	addMember
member	changeSubject
member	deleteMember
member	updateMember
membership	addMembership
membership	deleteMembership
membership	updateMembership
permission	permissionChangeOnRole (Grouper 2.1+)
privilege	addPrivilege
privilege	deletePrivilege
privilege	updatePrivilege
roleSet	addRoleSet
roleSet	deleteRoleSet
stem	addStem
stem	deleteStem
stem	updateStem

Implementing a consumer

Note: you should implement an EsbEventListener (layer on top), not a change log consumer

SQL View

There is a friendly SQL view: grouper_change_log_entry_v which will be more friendly to query if you are debugging something than the grouper_change_log_entry table

To do

- Add retry timeouts and max retries (currently it will just keep retrying every cycle of loader (every minute?) if there is an exception in target system. It should sleep for progressively longer and longer until hitting a max
- Add filters so that all systems are not notified on all change log entries
- Add hook on change log entry (pre/post insert/update/delete
- Add import/export to xml

Design / FAQ

- The change log is transactional. If a rollback or failure occurs in Grouper, the change log will be in sync.
- If the destination is unreachable, it will retry next time with the same record
- You can query the change log directly with the API through the DAO
- Change log entries are ordered so that they notify in an order that will work (i.e. create a group before adding members)
- There is a unique id (sequential sequence) for each row in change log
- The timestamp is in micros, and will never be the same on the same jvm as another record
- No database triggers are user, it's an all Java solution
- There are 12 columns to stash data in the change log entry
- Grouper keeps track of progress of consumers run in the loader

See also

Change Log Consumer in GSH