Daemon configuration

In Grouper v2.5+ there is re-organized configuration for daemon.

In the Grouper UI you can configure daemons on the UI "All daemons" screen.

This will assume you are storing configuration in the database, since the UI needs to save its settings

After a daemon is added/edited/deleted (if schedule is edited) then call the method to adjust quartz schedules

Daemon configuration in UI

Access the configurations in Miscellaneous Administration All daemons (existing screen)

Initial notes and limitations

If a daemon is running it might need to finish before picking up the new configuration

Add new daemon

The upper right drop-down should have an "Add daemon" option

The first option is a drop down to select the daemon type, should be Loader, Change log consumer, Messaging in, Messaging out, Generic daemon job

Daemon type	Notes
Loader	Show instructions for loader: 1. The loader configuration is on a group 2. For a "simple" loader, navigate to the group that should be loaded 3. For a "list" loader, navigate to a group which is not in the folder to be loaded 4. Click on the MoreLoader tab
Anything else	Show a wizard to configure options

Delete daemon

Each listing in the daemon table should add a new dropdown option "Delete daemon". This should take you to a confirmation screen.

Daemon type	Notes
Loader	The delete button will navigate to the edit screen of that daemon. There should be a note at the top of the screen that says "To delete this loader job set the "Loader" to "No, does not have loader configuration"
Other	Confirmation screen will ask if the user is sure, if they confirm, then delete any database configuration for this job. If there is "base" config settings then override them to blank

Edit daemon

Daemon type	Notes
Loader	Navigate to the edit screen of that daemon (note, there might be an example of this in the Misc Loader screen)

Other	Show the wizard to edit the daemon

Implementation

Each daemon class should be a simple javabean that can be constructed with a no-arg constructor

There should be a simple interface that helps identify the configuration customizer for the daemon

```
public interface HasGrouperDaemonConfiguration {
  public GrouperDaemonConfiguration getGrouperDaemonConfiguration();
}
```

The GrouperDaemonConfiguration class will be similar to the GrouperExternalSystem class in that Daemon configurations will extend the abstract GrouperDaemonConfiguration class and provide guidance about where configuration is stored and validated. There could be configuration inline with the daemon (in grouper-loader.properties) or it could be in other places too. The only time the "other" configuration should be displayed is if it has a one-to-one relationship with the daemon.

Identify the daemons and properties

Note, these lists will get out of date but they show the current configurations as of v2.5.24. These are just a few examples

Change log temp to change log

grouper-loader.properties

```
# should the change log temp to change log daemon run?  Note, this should be true
# {valueType: "boolean", required: true}
changeLog.changeLogTempToChangeLog.enable = true

#quartz cron-like schedule for change log temp to change log daemon, the default is 50 seconds after every
minute: 50 * * * * ?
# {valueType: "string"}
changeLog.changeLogTempToChangeLog.quartz.cron =

# The max number of changes to send to a change log consumer at one time
# {valueType: "integer", required: true}
changeLog.changeLogConsumerBatchSize = 1000

# Should the change log include flattened memberships?
# {valueType: "boolean", required: true}
changeLog.includeFlattenedMemberships = true

# Should the change log include flattened privileges?
# {valueType: "boolean", required: true}
changeLog.includeFlattenedPrivileges = true

# Should the change log include roles that have had permission changes?
# {valueType: "boolean", required: true}
changeLog.includeRolesWithPermissionChanges = false

# Should the change log include subjects that have had permission changes?
# {valueType: "boolean", required: true}
changeLog.includeSubjectsWithPermissionChanges = false

# Should the change log include non-flattened (immediate and composite only) memberships?
# {valueType: "boolean", required: true}
changeLog.includeNonFlattenedMemberships = false

# Should the change log include non-flattened (immediate only) privileges?
# {valueType: "boolean", required: true}
changeLog.includeNonFlattenedPrivileges = false

# Once the number of change log updates exceeds this value, the transaction will commit and a new one will be
created
# {valueType: "integer", required: true}
changeLog.tooManyChangeLogUpdatesSize = 10000
```

Other jobs

grouper-loader.properties

by default (if there is no configuration controller) show configs by regex like external systems, e.g.

```
# Find and fix scheduler issues class
# {valueType: "class", mustExtendClass: "edu.internet2.middleware.grouper.app.loader.OtherJobBase",
mustImplementInterface: "org.quartz.Job"}
otherJob.schedulerCheckDaemon.class = edu.internet2.middleware.grouper.app.loader.GrouperDaemonSchedulerCheck

# Find and fix scheduler issues cron
# {valueType: "string"}
otherJob.schedulerCheckDaemon.quartzCron = 25 0/30 * * * ?

# If there hasnt been a success in the last X minutes, then kick this off from thread (not from daemon).  Who
is watching the watcher?
# If this is -1, then do not run a watcher thread
# {valueType: "integer", defaultValue: "35"}
otherJob.schedulerCheckDaemon.maxMinutesSinceSuccess = 35

# If there has been a daemon run in the last X minutes, then dont run manually.  -1 to not include.  Note, if
maxMinutesSinceSuccess is -1, then
# this config will not be used
# {valueType: "integer", defaultValue: "15"}
otherJob.schedulerCheckDaemon.minMinutesSinceStarted = 15
```

Provisioner

grouper-loader.properties

```
pick configId for this schedule (or select if edit)
changeLog.consumer.<configId>.quartzCron
pick type of provisioner (e.g. ldap, sql, box, etc)
pick configId of the provisioner (to link to the provisioner)
additional properties?
```

SQL provisioner (example of provisioner)

grouper-loader.properties

```
################################
## Table sync jobs
## tableSync jobs should use class: edu.internet2.middleware.grouper.app.tableSync.TableSyncOtherJob
## and include a setting to point to the grouperClient config, if not same: otherJob.<otherJobName>.
grouperClientTableSyncConfigKey = key
## this is the subtype of job to run: otherJob.<otherJobName>.syncType = fullSyncFull
## (can be: fullSyncFull, fullSyncGroupings, fullSyncChangeFlag, incrementalAllColumns, incrementalPrimaryKey)
################################

# Object Type Job class
# {valueType: "class", mustExtendClass: "edu.internet2.middleware.grouper.app.loader.OtherJobBase",
mustImplementInterface: "org.quartz.Job"}
# otherJob.membershipSync.class = edu.internet2.middleware.grouper.app.tableSync.TableSyncOtherJob

# Object Type Job cron
# {valueType: "string"}
# otherJob.membershipSync.quartzCron = 0 0/30 * * * ?

# this is the key in the grouper.client.properties that represents this job
# {valueType: "string"}
# otherJob.membershipSync.grouperClientTableSyncConfigKey = memberships

# fullSyncFull, fullSyncGroupings, fullSyncChangeFlag, incrementalAllColumns, incrementalPrimaryKey
# {valueType: "string"}
# otherJob.membershipSync.syncType = fullSyncFull
```

LDAP provisioner full sync (not PSPNG)

grouper-loader.properties

Note this doesnt exist yet but this screen should be implemented anyways

```
################################
## LDAP provisioner full sync
################################

# Object Type Job class
# {valueType: "class", mustExtendClass: "edu.internet2.middleware.grouper.app.loader.OtherJobBase"}
# otherJob.ldapSync.class = edu.internet2.middleware.grouper.app.ldapProvisioning.LdapProvisioningOtherJob

# Object Type Job cron
# {valueType: "string"}
# otherJob.ldapSync.quartzCron = 0 0 3 * * ?

# this is the key in the grouper.client.properties that represents this job
# {valueType: "string"}
# otherJob.ldapSync.ldapProvisionerConfigId = myLdapSync

# fullSyncSynchronous, fullSyncAsynchronous, fullSyncDryRun
# {valueType: "string"}
# otherJob.ldapSync.syncType = fullSyncSynchronous
```

Rule daemon

grouper-loader.properties

```
# when the rules validations and daemons run.  Leave blank to not run
# {valueType: "string"}
rules.quartz.cron = 0 0 7 * * ?
```

Rule change log consumer

grouper-loader.properties

```
# rules consumer, needed for some of the Grouper rule types to run (e.g. flattenedMembershipRemove,
flattenedMembershipAdd)
# {valueType: "class", mustExtendClass: "edu.internet2.middleware.grouper.changeLog.ChangeLogConsumerBase"}
changeLog.consumer.grouperRules.class = edu.internet2.middleware.grouper.changeLog.esb.consumer.RuleConsumer

# rules consumer, needed for some of the Grouper rule types to run (e.g. flattenedMembershipRemove,
flattenedMembershipAdd)
# {valueType: "string"}
changeLog.consumer.grouperRules.quartzCron =
```

PSPNG

grouper-loader.properties

```
otherJob.pspng_oneprodFull_full.class = edu.internet2.middleware.grouper.pspng.FullSyncStarter
otherJob.pspng_oneprodFull_full.quartzCron = 0 10 20 ? * SAT *
```

Built in messaging daemon

grouper-loader.properties

```
###################################
## grouper builtin messaging cleanup cron
###################################

#quartz cron-like schedule for grouper messaging daemon.
#leave blank to disable this, the default is every hour, 10 minutes after the hour
#this daemon does cleanup on the builtin messaging table
# {valueType: "string"}
changeLog.builtinMessagingDaemon.quartz.cron = 0 10 * * * ?

# after three days of not consuming messages, delete them, if -1, dont run this daemon
# {valueType: "integer", required: true}
grouper.builtin.messaging.deleteAllMessagesMoreThanHoursOld = 72

# after three hours of having processed messages, delete them.  Note, if this is -1 just delete when marking
processed
# {valueType: "integer", required: true}
grouper.builtin.messaging.deleteProcessedMessagesMoreThanMinutesOld = 180
```

Enabled / disabled daemon

```
###################################
## enabled / disabled cron
###################################

#quartz cron-like schedule for enabled/disabled daemon.  Note, this has nothing to do with the changelog
#leave blank to disable this, the default is 12:01am, 11:01am, 3:01pm every day: 0 1 0,11,15 * * ?
# {valueType: "string"}
changeLog.enabledDisabled.quartz.cron = 0 1 0,11,15 * * ?
```

Grouper report

grouper-loader.properties

```
##################################
## Daily report
##################################
#quartz cron-like schedule for daily grouper report, the default is 7am every day: 0 0 7 * * ?
#leave blank to disable this
# {valueType: "string"}
daily.report.quartz.cron =

#comma separated email addresses to email the daily report, e.g. a@b.c, b@c.d
# {valueType: "string", multiple: true}
daily.report.emailTo =

#days on which usdu should run with daily report (comma separated)
#blank means run never.   e.g. to run on all days: monday, tuesday, wednesday, thursday, friday, saturday,
sunday
# {valueType: "string", multiple: true}
daily.report.usdu.daysToRun = monday, tuesday, wednesday, thursday, friday, saturday, sunday

#days on which bad membership finder should run with daily report (comma separated)
#blank means run never.   e.g. to run on all days: monday, tuesday, wednesday, thursday, friday, saturday,
sunday
# {valueType: "string", multiple: true}
daily.report.badMembership.daysToRun = monday, tuesday, wednesday, thursday, friday, saturday, sunday

#if you put a directory here, the daily reports will be saved there, and you can
#link up to a web service or store them or whatever.  e.g. /home/grouper/reports/
# {valueType: "string"}
daily.report.saveInDirectory =
```

Delete logs

(schedule is hard coded)

grouper-loader.properties

```
# number of days to retain db logs in table grouperloader_log.  -1 is forever.  default is 7
# {valueType: "integer", required: true}
loader.retain.db.logs.days=7

# number of days to retain db rows in grouper_change_log_entry.  -1 is forever.  default is 14
# {valueType: "integer", required: true}
loader.retain.db.change_log_entry.days=14
```

Find bad memberships

grouper-loader.properties

```
# Find and fix bad memberships class
# {valueType: "class", mustExtendClass: "edu.internet2.middleware.grouper.app.loader.OtherJobBase",
mustImplementInterface: "org.quartz.Job"}
otherJob.findBadMemberships.class = edu.internet2.middleware.grouper.misc.FindBadMembershipsDaemon

# Find and fix bad memberships cron
# {valueType: "string"}
otherJob.findBadMemberships.quartzCron = 0 0 1 * * ?
```

Messaging from ESB

```
#####################################
## Messaging integration with ESB, send change log entries to a messaging system
#####################################

# note, change "messagingEsb" in key to be the name of the consumer.  e.g. changeLog.consumer.myAzureConsumer.
class
# note, routingKey property is valid only for rabbitmq. For other messaging systems, it is ignored.
#changeLog.consumer.messagingEsb.class = edu.internet2.middleware.grouper.changeLog.esb.consumer.EsbConsumer

# quartz cron
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.quartzCron$"}
#changeLog.consumer.messagingEsb.quartzCron = 0 * * * * ?

# el filter
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.elfilter$"}
#changeLog.consumer.messagingEsb.elfilter = event.eventType eq 'GROUP_DELETE' || event.eventType eq 'GROUP_ADD'
|| event.eventType eq 'MEMBERSHIP_DELETE' || event.eventType eq 'MEMBERSHIP_ADD'

# publishing class
# {valueType: "class", mustExtendClass: "edu.internet2.middleware.grouper.changeLog.esb.consumer.
EsbMessagingPublisher", regex: "^changeLog\\.consumer\\.([^.]+)\\.publisher\\.class$"}
#changeLog.consumer.messagingEsb.publisher.class = edu.internet2.middleware.grouper.changeLog.esb.consumer.
EsbMessagingPublisher

# messaging system name
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.messagingSystemName$"}
#changeLog.consumer.messagingEsb.publisher.messagingSystemName = grouperBuiltinMessaging

# routing key
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.routingKey$"}
#changeLog.consumer.messagingEsb.publisher.routingKey =

# EL replacement definition. groupName is the variable for the name of the group. grouperUtil is the class
GrouperUtilElSafe can be used for utility methods.
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.regexRoutingKeyReplacementDefinition$"}
#changeLog.consumer.messagingEsb.regexRoutingKeyReplacementDefinition = ${groupName.replaceFirst('hawaii.edu',
'group.modify').replace(':enrolled', '').replace(':waitlisted', '').replace(':withdrawn', '')}

# replace routing key with periods
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.replaceRoutingKeyColonsWithPeriods$"}
#changeLog.consumer.messagingEsb.replaceRoutingKeyColonsWithPeriods = true

# queue or topic
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.publisher\\.messageQueueType$"}
#changeLog.consumer.messagingEsb.publisher.messageQueueType = queue

# queue or topic name
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.publisher\\.queueOrTopicName$"}
#changeLog.consumer.messagingEsb.publisher.queueOrTopicName = abc

# exchange type for rabbitmq. valid options are DIRECT, TOPIC, HEADERS, FANOUT
# {valueType: "string", regex: "^changeLog\\.consumer\\.([^.]+)\\.publisher\\.exchangeType$"}
#changeLog.consumer.messagingEsb.publisher.exchangeType =
```

Incremental loader

grouper-loader.properties

```
############################
## Incremental loader jobs
############################

# incremental loader job class
# {valueType: "class", regex: "^otherJob.([^.]+).class$", mustExtendClass: "edu.internet2.middleware.grouper.
app.loader.GrouperLoaderIncrementalJob"}
# otherJob.incrementalLoader1.class = edu.internet2.middleware.grouper.app.loader.GrouperLoaderIncrementalJob

# incremental loader job cron
# {valueType: "string", regex: "^otherJob.([^.]+).quartzCron$"}
# otherJob.incrementalLoader1.quartzCron = 0 * * * * ?

# incremental loader job database name
# {valueType: "string", regex: "^otherJob.([^.]+).databaseName$"}
# otherJob.incrementalLoader1.databaseName=warehouse

# incremental loader job table name
# {valueType: "string", regex: "^otherJob.([^.]+).tableName$"}
# otherJob.incrementalLoader1.tableName=myincrementaltable

# incremental loader full sync threshold
# If there are more than this many changes for a single loader job, then invoke the full sync instead.  This
could improve performance but also handle fail safe which isn't part of the incremental sync.
# {valueType: "integer", regex: "^otherJob.([^.]+).fullSyncThreshold$"}
# otherJob.incrementalLoader1.fullSyncThreshold=100

# whether subject lookups in the data source should be case insensitive.  only applicable for sql loader jobs.
note, if true, for some databases (e.g. oracle), you may need a function based index in your data source for
the function "lower" for better performance
# {valueType: "boolean", regex: "^otherJob.([^.]+).caseInsensitiveSubjectLookupsInDataSource$"}
# otherJob.incrementalLoader1.caseInsensitiveSubjectLookupsInDataSource=false
```

CSV reports

grouper-loader.properties

```
#######################################################
## CSV reports
## "reportId" is the key of the config, change that for your csv report
#######################################################


# set this to enable the instrumentation
# {valueType: "class", mustExtendClass: "edu.internet2.middleware.grouper.app.loader.OtherJobBase"}
# otherJob.reportId.class = edu.internet2.middleware.grouper.app.reports.GrouperCsvReportJob

# cron string
# {valueType: "string"}
# otherJob.reportId.quartzCron = 0 21 7 * * ?

# query to run
# {valueType: "string", regex: "^otherJob\\.([^.]+)\\.csvReport\\.query$"}
# otherJob.reportId.csvReport.query = select USER_ID, USER_NAME, EMAIL_ADDRESS, AUTH_TYPE, TITLE, DEPARTMENT,
CUSTOM_STRING, DAY_PASS, CUSTOM_STRING2, GROUPS from some_view

# database to hit
# {valueType: "string", regex: "^otherJob\\.([^.]+)\\.csvReport\\.database$"}
# otherJob.reportId.csvReport.database = pennCommunity

# remove underscores and capitalize headers, go from USER_NAME to UserName
# {valueType: "string", regex: "^otherJob\\.([^.]+)\\.csvReport\\.removeUnderscoresAndCapitalizeHeaders$"}
# otherJob.reportId.csvReport.removeUnderscoresAndCapitalizeHeaders = false

# fileName, e.g. myFile.csv or /opt/whatever/myFile.csv.  If blank will create a name
# {valueType: "string", regex: "^otherJob\\.([^.]+)\\.csvReport\\.database$"}
# otherJob.reportId.csvReport.fileName = MyFile.csv

# sftp config id (from grouper.properties) if sftp'ing this file somewhere, otherwise blank
# https://spaces.at.internet2.edu/display/Grouper/Grouper+Sftp+files
# {valueType: "string", regex: "^otherJob\\.([^.]+)\\.csvReport\\.sftp\\.configId$"}
# otherJob.reportId.csvReport.sftp.configId = someSftpServer

# remote file to sftp to if sftp'ing
# {valueType: "string", regex: "^otherJob\\.([^.]+)\\.csvReport\\.sftp\\.fileNameRemote$"}
# otherJob.reportId.csvReport.sftp.fileNameRemote = /data01/whatever/MyFile.csv

# if the file should be deleted from the grouper daemon server after sending it
# {valueType: "boolean", regex: "^otherJob\\.([^.]+)\\.csvReport\\.deleteFile$"}
# otherJob.reportId.csvReport.deleteFile = true
```