Grouper instrumentation

The goal of this project is to centrally collect TIER data about Grouper deployment to help improve Grouper and give information to TIER constituents about Grouper usage. As of Grouper 2.3 Patch 62, no information is automatically sent to TIER as we are waiting for TIER to have a production collector. However, the Grouper UI has a new feature that allows local administrators to view some of the data that would be sent to TIER.

Overall Design

- Grouper has a central database which can store information for a Grouper env at an institution
- Each JVM process (currently daemon, UI, WS) can periodically check in to the DB (e.g. every 1 hour)
    - Let it know its UUID, type of process, number of tx of various types since last checkin, etc
- Daily a new TIER instrumentation daemon could collate information in the database, and glean other information (e.g. is PSP or PSPNG running) and after consulting the discovery service, send a report to the TIER collector. This is currently not automatically enabled as we are waiting for a production TIER collector.

Instrumentation Thread

A new thread runs in each JVM process that keeps track of updates being made of various types:

- Grouper UI servlet requests
- Grouper WS servlet requests
- Group adds
- Group deletes
- Folder adds
- Folder deletes
- Membership adds
- Membership deletes
- (more later)

The thread will update counts of these various types to the Grouper database every 1 hour by default (configured via grouper.properties - instrumentation.updateGrouperIntervalInSeconds).

Another configuration option will specify the increments to keep counts of (instrumentation.updateIncrements). E.g. if we're keeping counts by 10 minutes or hour or day. This also defaults to 1 hour.

This data in Grouper is kept for 30 days but configurable (instrumentation.retainData.days).

Grouper UI

Grouper administrators can view the counts gathered by the instrumentation thread in the Grouper UI. Other users can also be given access by configuring uiV2.admin.instrumentation.must.be.in.group.

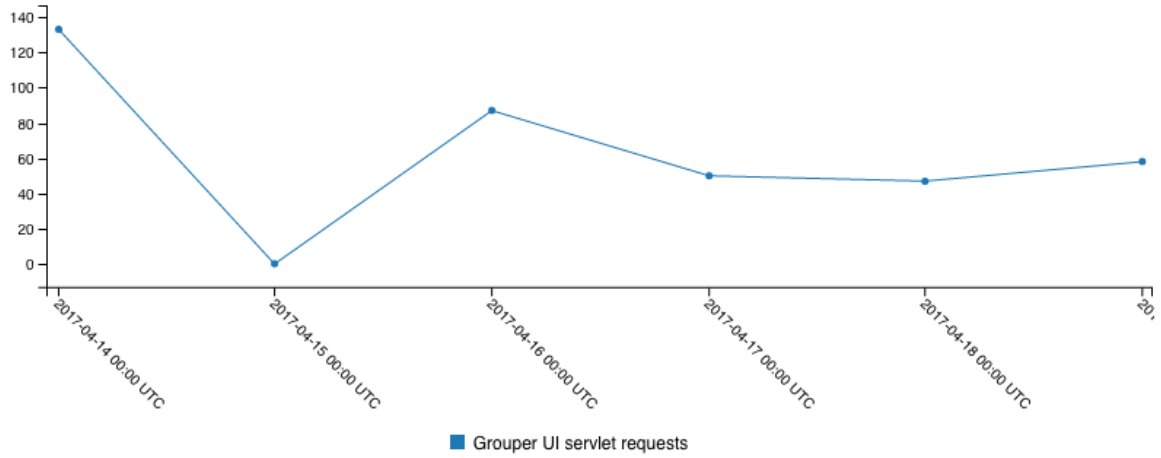
Under Quick Links, click on Miscellaneous and then click on Instrumentation.

At the top, it will show you each of your Grouper instances (currently Daemon, UI, and WS).

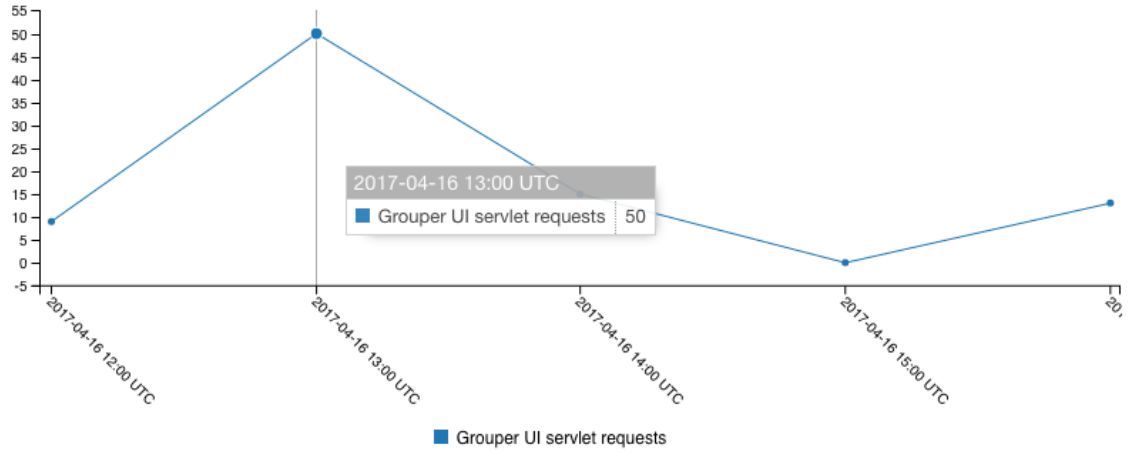
UUID	Engine name	Server label	Last update
1e91784c28de463389471c4aa0a25192	grouperUI	i2midev6.internet2.edu	Thu Apr 20 13:28:32 UTC 2017
f87eeecf3fd84d668944580c7e10f6c0	grouperLoader	i2midev6.internet2.edu	Thu Apr 20 13:27:38 UTC 2017

The Last update column indicates the last time that the instrumentation thread for that instance updated the database with its counts.

Below that, you can view the overall counts for the various types.



The counts are displayed by day. If you select a specific day, you can view hourly counts for that day.



Also, you can view the same data for each individual instance by clicking on the instance UUID.

Enable TIER collection

Note this will go to a test collector. Shouldn't be used in production at the moment.

Get patches for 2.3 (24 and 25)

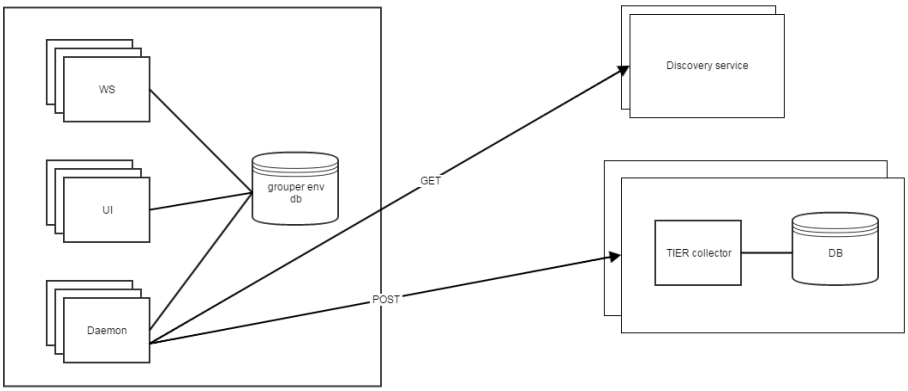
Set this in grouper-loader.properties

```
otherJob.tierInstrumentationDaemon.class = edu.internet2.middleware.grouper.instrumentation.
TierInstrumentationDaemon
otherJob.tierInstrumentationDaemon.quartzCron = 0 0 2 * * ?
```

Technical Details

Discovery service

- Simple static HTTP resource(s) that would be always/forever available (e.g. hosted at AWS?) to designate where the TIER collector(s) are



- Note, we can start out with one collector. Client failover is optional, the first endpoint might be used only for simple clients
- e.g. request: GET (current)
  - https://id.internet2.edu/ti/jrd/collector
  - results: http://cerif.org/elk /
- e.g. request: GET (outdated)
  - https://s3.amazonaws.com/edu.internet2.tierinstrumentationcollector.discovery0/tierInstrumentationCollector_discovery.json
  - https://s3-us-west-1.amazonaws.com/edu.internet2.tierinstrumentationcollector.discovery1/tierInstrumentationCollector_discovery.json

```
{
  serviceEnabled: true,
  endpoints: [
    {
       uri: "https://grouperdemo.internet2.edu/tierInstrumentationCollector/tierInstrumentationCollector/v1
/upload"
    },
    {
       uri: "https://grouperdemo2.internet2.edu/tierInstrumentationCollector2/tierInstrumentationCollector/v1
/upload"
    }

  ]
}
```

Collector

- Simple REST endpoint that takes any name/value pairs in JSON in a simple structure of single valued strings
- The collector can just store each resource it gets and doesnt care what the attributes are, so the components can change their data as they need
- Of course the reporting and processing needs to take the attributes and values into account
- e.g. submission: POST https://tiercollector1.internet2.edu/v1/collector/dailyReport

```
{
   "reportFormat":1,
   "component":"grouper",
   "institution”:”Penn”,
   "environment”:”prod”,
   "version":"2.3.0",
   "platformWindows":false,
   "platformLinux":false,
   "platformMac":true,
   "platformSolaris":false,
   "transactionCountMemberships":6,
   "transactionCountPrivileges":0,
   "registryCountDirectMemberships":2,
   "registryCountDirectPrivileges":54,
   "registryCountDirectPermissions":0,
   "provisionToLdapUsingPsp":false,
   "provisionToLdapUsingPspng":false,
   "patchesInstalled":[

   ],
   "instances":[
```

```
{
   "uuid":"d0a363444b49489591b80690a6edc09d",
   "engineName":"grouperShell",
   "serverLabel":"-2",
   "lastUpdate":1489762214975,
   "newCounts":[

   ]
},
{
   "uuid":"2898de277109417ebccd00cee00f0649",
   "engineName":"grouperLoader",
   "serverLabel":"-2",
   "lastUpdate":1489769384218,
   "newCounts":[

   ]
},
{
   "uuid":"1a8885db0b564e94b3958dbfc7b032f0",
   "engineName":"grouperUI",
   "serverLabel":"-2",
   "lastUpdate":1489769384725,
   "newCounts":[
      {
         "startTime":1489767550000,
         "duration":5000,
         "UI_REQUESTS":5
      },
      {
         "startTime":1489767600000,
         "duration":5000,
         "UI_REQUESTS":1
      },
      {
         "startTime":1489767635000,
         "duration":5000,
         "UI_REQUESTS":1
      },
      {
         "startTime":1489767650000,
         "duration":5000,
         "UI_REQUESTS":1
      },
      {
         "startTime":1489767660000,
         "duration":5000,
         "API_GROUP_ADD":1,
         "UI_REQUESTS":2
      },
      {
         "startTime":1489767690000,
         "duration":5000,
         "API_GROUP_DELETE":1,
         "UI_REQUESTS":3
      },
      {
         "startTime":1489767720000,
         "duration":5000,
         "UI_REQUESTS":1
      },
      {
         "startTime":1489767725000,
         "duration":5000,
         "UI_REQUESTS":3
      },
      {
         "startTime":1489767730000,
         "duration":5000,
         "UI_REQUESTS":3,
         "API_STEM_ADD":1
```

```
},
{
   "startTime":1489767735000,
   "duration":5000,
   "UI_REQUESTS":2,
   "API_STEM_DELETE":1
},
{
   "startTime":1489767770000,
   "duration":5000,
   "UI_REQUESTS":1
},
{
   "startTime":1489767790000,
   "duration":5000,
   "UI_REQUESTS":2
},
{
   "startTime":1489767795000,
   "duration":5000,
   "UI_REQUESTS":2,
   "API_GROUP_ADD":1
},
{
   "startTime":1489767800000,
   "duration":5000,
   "UI_REQUESTS":1
},
{
   "startTime":1489767805000,
   "duration":5000,
   "API_MEMBERSHIP_ADD":1,
   "UI_REQUESTS":3
},
{
   "startTime":1489767835000,
   "duration":5000,
   "UI_REQUESTS":1
},
{
   "startTime":1489769030000,
   "duration":5000,
   "UI_REQUESTS":4
},
{
   "startTime":1489769035000,
   "duration":5000,
   "UI_REQUESTS":1
},
{
   "startTime":1489769045000,
   "duration":5000,
   "UI_REQUESTS":4
},
{
   "startTime":1489769170000,
   "duration":5000,
   "UI_REQUESTS":1
},
{
   "startTime":1489769175000,
   "duration":5000,
   "UI_REQUESTS":2
},
{
   "startTime":1489769275000,
   "duration":5000,
   "API_MEMBERSHIP_DELETE":1,
   "UI_REQUESTS":1
},
{
```

```
            "startTime":1489769315000,
            "duration":5000,
            "API_MEMBERSHIP_ADD":1,
            "UI_REQUESTS":4
         }
      ]
   }
]
}
```

Collecting Counts

- Enabled for UI, WS and Daemon
- Data will be kept in the folder etc:attribute:instrumentationData
- Collect counts of servlet requests, group adds/deletes, membership adds/deletes, folders adds/deletes
- UI and WS can start a new thread when the servlet first initializes, Daemon can start a new thread when it starts.
- The new thread (a single thread executor) will enable stat collection (i.e. set some static variable)
- Grouper api and ui code will update various static lists of timestamps indicating when each operation is done
- A config option will determine how often the thread will go through the timestamps in memory and update the grouper database. Lower means fewer gaps when the process is killed.
- Another config option will specify the increments to keep counts of. E.g. if we're keeping counts by 10 minutes or hour or day.
- When the UI thread starts up, check to see if an "<ENGINE_NAME>_instrumentation.dat" file exists. This file will contain the uuid of this instance. The location of the *_instrumentation.dat file(s) can be optionally defined in grouper.properties value *instrumentation.instanceFile.directory*. If not defined, the default is to read from or create in the log directory.
- If *_instrumentation.dat doesn't exist, create it and create a corresponding attribute in grouper, e.g. etc:attribute:instrumentationData:instrumentationDataInstances:theuuid (def = etc:attribute:instrumentationData:instrumentationDataInstancesDef)
- The <ENGINE_NAME>_instrumentation.dat file should have a trivial update whenever the thread flushes to the database just in case the system is cleaning old files.
- There will be a group used for assignments - etc:attribute:instrumentationData:instrumentationDataInstancesGroup.
- There will be a single assign multi valued attribute - etc:attribute:instrumentationData:instrumentationDataInstanceCounts (def = etc:attribute:instrumentationData:instrumentationDataInstanceCountsDef)
- There will also be other attributes (def = etc:attribute:instrumentationData:instrumentationDataInstanceDetailsDef) - etc:attribute:instrumentationData:instrumentationDataInstanceLastUpdate, etc:attribute:instrumentationData:instrumentationDataInstanceEngineName, etc:attribute:instrumentationData:instrumentationDataInstanceServerLabel
- So etc:attribute:instrumentationData:instrumentationDataInstances:theuuid will be assigned to etc:attribute:instrumentationData:instrumentationDataInstancesGroup. And on that assignment will live assignments with actual data (instrumentationDataInstanceCounts, instrumentationDataInstanceLastUpdate, instrumentationDataInstanceEngineName, instrumentationDataInstanceServerLabel)
- The value of the assignment on the assignment (instrumentationDataCounts) will be like:

```
{"startTime" : 1486753200000, "duration" : 600000, "UI_REQUESTS" : 30, "API_GROUP_ADD" : 5,
"API_GROUP_DELETE" : 3}
```

- There may be multiple values added each time it runs. For example, if the database is updated every hour and the increment is every 10 minutes, then it could add 6 of these.

```
{"startTime" : 1486753200000, "duration" : 600000, "UI_REQUESTS" : 30, "API_GROUP_ADD" : 5,
"API_GROUP_DELETE" : 3}
{"startTime" : 1486753800000, "duration" : 600000, "UI_REQUESTS" : 300, "API_GROUP_ADD" : 2,
"API_GROUP_DELETE" : 6}
{"startTime" : 1486754400000, "duration" : 600000, "UI_REQUESTS" : 3000, "API_GROUP_ADD" : 1,
"API_GROUP_DELETE" : 2}etc
```

- The TIER instrumentation daemon will sends these to TIER.

```
"instances" : [ { "uuid" : "uuid1",
                  "engineName" : "grouperUI",
                  "serverLabel" : "ui-01"
                  "lastUpdate" : 1488825739828,
                  "newCounts" : [{"startTime" : 1486753200000, "duration" : 600000, "UI_REQUESTS" : 30,
"API_GROUP_ADD" : 5, "API_GROUP_DELETE" : 3},
                                 {"startTime" : 1486753800000, "duration" : 600000, "UI_REQUESTS" : 300,
"API_GROUP_ADD" : 2, "API_GROUP_DELETE" : 6},
                                 {"startTime" : 1486754400000, "duration" : 600000, "UI_REQUESTS" :
3000, "API_GROUP_ADD" : 1, "API_GROUP_DELETE" : 2}]
                },
                { "uuid" : "uuid2",
                  "serverLabel" : "ui-02"
                  "engineName" : "grouperUI",
                  "lastUpdate" : 1488825739829
                },
                { "uuid" : "uuid3",
                  "serverLabel" : "ws-01"
                  "engineName" : "grouperWS",
                  "lastUpdate" : 1488825739829
                },
                { "uuid" : "uuid4",
                  "serverLabel" : "ws-02"
                  "engineName" : "grouperWS",
                  "lastUpdate" : 1488825739829
                },
                { "uuid" : "uuid5",
                  "serverLabel" : "daemon-01"
                  "engineName" : "grouperLoader",
                  "lastUpdate" : 1488825739829
                }
              ]
```

- An attribute will be created for each collector (e.g. etc:attribute:instrumentationData:instrumentationDataCollectors: OTHER_JOB_tierInstrumentationDaemon). This will be assigned to another group (etc:attribute:instrumentationData: instrumentationDataCollectorsGroup). And that assignment will have the time the collector was last updated (etc:attribute:instrumentationData: instrumentationDataCollectorLastUpdate).
- The values won't be audited (user audit or point in time audit)
- The cleanLogs daemon will delete counts older than 30 days (configurable).
- Adding more counts
    - There is a built in enum of types - InstrumentationDataBuiltinTypes
    - You can also dynamically configure other types when starting the instrumentation thread.
    - InstrumentationThread.startThread(GrouperContext.retrieveDefaultContext().getGrouperEngine(), customTypes);
    - Maybe there needs to be a way to add more types after startup so that the PSP (or something else) can add to it?

Run daemon from GSH

```
-- DOES THIS WORK????
loaderRunOneJob("OTHER_JOB_tierInstrumentationDaemon");
```

Notes

- Keith is interested in LogStash
- Scott is interested in Metrics (java library)