Grouper to ESB design

This page describes the design of the link for sending data from Grouper to the ESB (Grouper->ESB). The design for the reverse data flow (ESB->Grouper) will follow the successful prototyping of the Grouper->ESB link.

Existing infrastructure

Class to which notifications are sent must extend abstract class edu.internet2.middleware.grouper.changeLog.ChangeLogConsumerBase

Notification is executed by a quartz job. This job is of type edu.internet2.middleware.grouper.app.loader.GrouperLoaderType which hooks the configured ChangeLogConsumer into the CHANGE_LOG job type execution.

The quartz job is scheduled in edu.internet2.middleware.grouper.app.loader.GrouperLoader (which has a public static void main method for starting Grouper as a daemon process).

Proposed design of EBS notifications

ESB notfications (Grouper -> ESB) will be implemented as a ChangeLogEsbConsumer class extending ChangeLogConsumer in package edu.internet2.middleware.grouper.changeLog.esb, configured in grouper-loader.properties.

The class will package each notification in JSON format. It will send it to a class which will extend an abstract class ChangeLogEsbPublisherBase, and be hooked in through configuration in grouper.properties (or grouper-esb.properties if things look like getting verbose). The class which extents ChangeLogEsbPublisherBase will implement all the logic to send the notification to an ESB. Initially HTTP(S) will be supported using the Apache Commons HTTP client, but it will be easy to add another class with a different communications protocol. The class can send the notfication in JSON format, or convert it to another format (such as XML). All required parameters will be loaded from grouper-loader.properties.

A simple bean has been added with a common format for packaging events from grouper. Values available with events are resolved to meaningful names, which are then added to properties in the bean This is converted to JSON using libraries already included in the Grouper binary distribution

Events can be filtered in the consumer using EL on the bean.

Outstanding questions

- How much information about groups and subjects will the ESB require for operations? Additional information on subjects will be available if configured.
- What to use to convert to XML? I favour Castor as then mappings can be configured to give control over the XML written. I also favour converting from JSON to XML, rather than from the Java class as this produces the simplest data structure (which can be enriched using Castor mappings).

Prototype performance

A prototype with this design has been tested using the following setup:

- Grouper 1.5 running on my laptop (in Eclipse)
- Postgres running on my laptop, hosting both the grouper database and the subject source
- Mule ESB running on my laptop (in the same instance of Eclipse as Grouper)
- An ldap directory (eDirectory) running in a VM (not on my laptop!), comms over wi-fi

A realistic task of syncing membership adds to the LDAP directory through the ESB was set up. Events are raised by Grouper and sent as JSON over HTTP to the ESB. The ESB resolves the name of the subject to the full LDAP dn of the associated object in LDAP with a search, and adds an attribute value to the object in the directory.

With this setup, running 1500 membership adds performance was 19 operations/second. For membership removals, performance was faster at 25 operations/second (which is probably accounted for by the LDAP directory). This is fast enough for me.

Laptop specs:

dual coreprocessor, model name : Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz
MemTotal: 2820264 kB
OS: Linux 2.6.33
Postgres: 8.4
Mule: 2.2.1