Prototype DieselNet Handler Module

Contents

1. Introduction

An overview of a prototype DieselNet Handler module for ORCA has been completed by Brian Lynn.

There are three files attached:
dieselnet-handler.tar.gz - Contains all of the files required to build the DieselNet handler for ORCA.
files.tar.gz - Contains copies of some ORCA files that were modified to support the addition of the DieselNet handler.
dome-rpcserver.tar.gz - This is the source of the XML-RPC server that executes on the DOME server. The functions invoked by join and leave are placeholders in this version.

2. Summary

Provided by Brian Lynn on May 13, 2009:
Attached is the complement to the DieselNet controller: the DieselNet handler.

Whereas the controller deals with receiving requests for leases and inserting those requests into ORCA, the handler takes action on resources when a lease is granted (join) or revoked (leave) by ORCA. When the lease begins ORCA executes a DOME-supplied Apache ant script with "join" as the target; when the lease expires the script is executed with "leave" as the target. The join and leave targets execute a DOME-specific Apache ant task (see http://ant.apache.org/manual/tutorial-writing-tasks.html), passing to the task the opaque properties set by the controller. For DOME, the tasks simply send XML-RPC messages to an XML-RPC server executing on a DOME system. The DOME server is then responsible for communicating leases to the buses.

I have included:

dieselnet-handler.tar.gz - Contains all of the files required to build the DieselNet handler for ORCA.

files.tar.gz - Contains copies of some ORCA files that were modified to support the addition of the DieselNet handler.
dome-rpcserver.tar.gz - This is the source of the XML-RPC server that executes on the DOME server. The functions invoked by join and leave are placeholders in this version.

The details are below. Once again, thanks to David.

dieselnet-handler.tar.gz
We build the DieselNet handler as part of the ORCA WAR, as opposed to installing it as a package as is done for the controller. To install and build the handler:

```
$ cd $ORCA_HOME/handlers
$ tar zxf dieselnet-handler.tar.gz
$ cd dieselnet
$ mvn clean
$ mvn install
```

The scripts and build files in this directory were derived from the ec2 handler. If you look in directory resources/handlers/dieselnet you'll see the join/leave ant script files (start with handler.xml).

The source directory (src/main/java/orca/handlers/dieselnet) contains:
DieselnetTask - Abstract class that extends the Apache ant Task class. This contains methods for setting properties defined by the ant script. It also contains the execute() method where the Task thread begins.
DieselnetTaskJoin, DieselnetLeave - These extend DieselnetTask and are invoked by the respective ant script join and leave targets.
DieselnetClient - Helper class for setting up the XML-RPC client functions.

In the ant script (handler.xml) you'll see something like this:

```
<dieselnet.join
  domeExperiment="${domeExperiment}"
  domeInstance="${domeInstance}"
  domeUser="${domeUser}"
  domeTicketId="${domeTicketId}"
/>
```

This is where the properties that had been associated with the lease are passed to the task. The task object is defined in dieselnet.tasks.xml.

Like the controller, the handler needs its own guid. The guid is set in resources/package.xml. It is also used in a configuration file (see below).

files.tar.gz
This tar contains various ORCA files that were modified to support DOME. To update ORCA:

```
   $ tar zxf files.tar.gz
   $ cd tmpmisc
   $ [backup existing handlers/build.xml]
   $ cp build.xml $ORCA_HOME/handlers/.
   $ [backup existing handlers/pom.xml]
   $ cp pom.xml $ORCA_HOME/handlers/.
   $ [backup existing webapp/maven_build.xml]
   $ cp maven_build.xml $ORCA_HOME/webapp/.
   $ [backup existing webapp/local/config.xml]
   $ cp config.xml $ORCA_HOME/webapp/local/.
```

A brief description of the modifications:

- handlers/build.xml - Adds the dieselnet directory to the remove.all target (todo: add "get" functions for checking files out of svn).
- handlers/pom.xml - Adds the dieselnet module.
- webapp/maven_build.xml - Adds the DieselNet handler to the ORCA WAR file that is built.
- webapp/local/config.xml - This is where we tell ORCA to invoke our handler.

We modify the webapp/local/config.xml file so that the DieselNet handler is invoked rather than the standard handler. Replacing the standard guid with the DieselNet guid does this. See line 70:

```
   <property name="resource.pool.handler.packageId.0"
   value="f7400992-1f30-4c3a-8134-9e00f780fc22" />
```

Another change that we made is that we wanted the DOME testbed to be represented as 1 unit (the default had been 3 units). See line 66:

```
   <property name="resource.pool.ticket.units.0"
   value="1" />
```

and line 206:

```
  <units>1</units>
```

Building ORCA with the Handler

Assuming the above changes were applied to a previously built ORCA, the following should allow you to update the orca.war file that gets deployed.

```
$ cd $ORCA_HOME/handlers/diselnet
$ mvn clean ; mvn install
$ cd $ORCA_HOME/webapp
$ mvn clean ; mvn package
$ cd $CATALINA_HOME/tomcat/webapp
$ mv orca.war orca.war.orig
$ rm -rf orca
$ cp $ORCA_HOME/webapp/target/orca.war .
```

You can now start tomcat and log in to the ORCA admin site.
If you click on the admin tab, both the View Handlers and View Packages options should indicate that the handler is installed. The DieselNet controller can then be installed and started. ORCA should now be ready.

dome-rpcserver.tar.gz
This is where the DOME-specific code for dealing with joins and leaves will reside. It receives the XML-RPC calls initiated by the ORCA ant script. For DOME, they will result in MySQL calls. Other scripts exist for the buses to periodically call and check the DB. (Due to the nature of connections to the buses, we pull things from a server to the buses, rather than push to the buses.)

To build and run the XML-RPC server:

```
$ tar zxf dome-rpcserver.tar.gz
$ cd DOME_RPC_Server
$ ant jar
$ cd run
$ java -jar dome-rpcserver.jar
```

As this point everything should be set up.
1) The python script (prior email) will use XML-RPC to invoke the controller.
2) The controller will request a lease from ORCA.
3) When ORCA grants the lease the ant script with a target of "join" will be executed.
4) The java Task is executed, which uses XML-RPC to contact the DOME RPC server.
5) When the lease expires ORCA executes the ant script with the "leave" target.
6) Another java Task is executed, which also uses XML-RPC to contact the DOME RPC server.

The output of the ant script executing join and leave can be seen in $CATALINA_HOME/logs/catalina.out.

- Brian