GROUNDWORK
OPEN SOURCE SOLUTIONS

Developing Feeders and Adapters for Foundation 1.1

Last Update: February, 9 2006 – Roger Ruttimann

Table of Contents

Foundation 1.1 Adapter development

The following document describes step by step how to develop a new feeder and adapter for inserting custom data streams into the Foundation framework.

The input source will be SNMP traps.

Development environment

The adapters are distributed as Java libraries (jar) and therefore you need the JAVA SDK available for compiling and packaging the adapters.
The tutorial uses ANT and MAVEN as build tools. You can get ANT and Maven binaries from the Apache site.
The tutorial uses the Foundation 1.1 source distribution which is available form http://sourceforge.net/projects/gwfoundation

Before you start

Install and configure the build tools and Foundation 1.1. Make sure that foundations build runs without errors.

Defining the data that needs to be integrated

As mentioned before the following example integrates (collects/normalizes) SNMP trap events.

What's the Application scope?

SNMP trap are treated as separate application and therefore we have to create a new ApplicationType: SNMPTRAP

What's the Entity Type?

SNMP trap messages will be stored in the LogMessage table and therefore the EntityType is LOG_MESSGAE

What fields need to be stored?

Define the attributes that are generated by the application and specify the if required and the default values if not required.

Attribute	***Type***	***Property***	***required / default value***
Host	String	no	required
Severity	Class Severity	no	required
IpAddress	String	yes	required
MonitorStatus	Class Severity	no	same as Severity
ReportDate	Date	no	default set at time inserted
LastInsertDate	Date	no	default set at time inserted. Can be set by SNMP_LOG message
Event_OID_numeric	String	yes	not required, default to null
Event_OID_symbolic	String	yes	not required, default to null
Event_Name	String	yes	not required, default to null
Category	String	yes	not required, default to null
Variable_Bindings	String	yes	not required, default to null
TextMessage	String	no	not required, default to null

Select fields that are application specific?
A list of fields need to be defined that are properties attached to the table defined as the Entity Type. The properties are application specific and are not part of the base table.
For SNMP traps the following fields and types will be stored:

PropertyName	***Type***
ipaddress	String
Event_OID_numeric	String
Event_OID_symbolic	String
Event_Name	String
Category	String
Variable_Bindings	String

Select fields for data consolidation?
This feature reduces to number of equal messages in the LogMessage table. For each insert the consolidation criteria will be applied to the incoming message as long as the flag in the XML stream is set to to a consolidation criteria name (consolidate="SNMPTRAP"). By default no consolidation criteria is applied. If the consolidation criteria matches the message counter for an existing message will increased and the date fields will be updated as following:

Field	***change***
FirstInserDate	unchanged
LastInsertDate	ReportDate
ReportDate	System (current time)

Fields that have to match before a message gets consolidated:

- OperationStatus, Host, Severity, ipaddress, MonitorStatus, Event_OID_numeric, Event_Name, Category, Variable_Bindings

Once the data set is defined and properties and consolidation criteria are defined the next step will be to update the database Meta data and writing an adapter for SNMP data normalization.

Database updates

The following database updates can be integrated into one database script that ships with the new adapter. It' always a good idea to check for the existence of an entry before attempting to insert in. For readability of the tutorial these steps are documented.

Since a new application was created the Application needs to be added to the database:
INSERT INTO ApplicationType(Name, Description) VALUES ("SNMPTRAP", "SNMP Trap application");

The new properties need to be created and assigned to the ApplicationType and EntityType

```
INSERT INTO PropertyType(Name, Description, isString)  VALUES ("ipaddress", "ipdddress of snmp device", 1);
INSERT INTO PropertyType(Name, Description, isString)  VALUES ("Event_OID_numeric", "Event_OID_numeric", 1);
and so on for all properties...

INSERT INTO ApplicationEntityProperty(ApplicationTypeID, EntityTypeID, PropertyTypeID, SortOrder) VALUES ((SELECT
ApplicationTypeID FROM ApplicationType WHERE Name='SNMPTRAP'),(SELECT EntityTypeID FROM EntityType WHERE
Name='LOG_MESSGE'),(SELECT PropertyTypeID FROM PropertyType WHERE Name = 'ipaddress'), 1);

INSERT INTO ApplicationEntityProperty(ApplicationTypeID, EntityTypeID, PropertyTypeID, SortOrder) VALUES ((SELECT
ApplicationTypeID FROM ApplicationType WHERE Name='SNMPTRAP'),(SELECT EntityTypeID FROM EntityType WHERE
Name='LOG_MESSGE'),(SELECT PropertyTypeID FROM PropertyType WHERE Name = 'Event_OID_numeric'), 1);
and so on for all properties...
```

The consolidation criteria needs to be named and the criteria of matching fields is a semi colon separated list

```
INSERT INTO ConsolidationCriteria(Name, Criteria)  VALUES ("SNMPTRAP", "OperationStatus;Host;Severity;ipaddress,
MonitorStatus; Event_OID_numeric;Event_Name;Category;Variable_Bindings");
```

Writing the Feeder

The feeder captures SNMP trap and send xml formated messages where all the fields to monitor are XML attributes to the Foundation listener component. The Listener listens on a configurable socket for incoming messages. The default is set to port 4913.

For SNMP traps the XML messages looks as following:

```
<SNMPTRAP
   MonitorServerName="localhost"
   Host="cisco2900.itgroundwork.com"
   Severity="Normal"
   MonitorStatus="Normal"
   ReportDate=""2005-10-25 04:20.44"
  LastInsertDate="2005-10-25 04:20.44"
   ipaddress="192.168.2.203"
  Event_OID_numeric=".1.3.6.1.4.1.9.0.1"
   Event_OID_symbolic="enterprises.9.0.1"
  Event_Name="tcpConnectionClose"
  Category="Status Events"
  Variable_Bindings="enterprises.9.2.9.3.1.1.1.1:5 tcpConnState.192.168.2.203.23.192.168.2.249.38591:synReceived
enterprises.9.2.6.1.1.5.192.168.2.203.23.192.168.2.249.38591:600
enterprises.9.2.6.1.1.1.192.168.2.203.23.192.168.2.249.38591:70
enterprises.9.2.6.1.1.2.192.168.2.203.23.192.168.2.249.38591:101 enterprises.9.2.9.2.1.18.1:"
  TextMessage="A tty trap signifies that a TCP connection, previously established with the sending protocol entity for the purposes
of a tty session, has been terminated.   5 synReceived 600 70 101 " />
```

Writing an adapter

Creating the Java project

- Inside the expanded Foundation package go into collagefeeder/adapter and create a new folder called snmp.
- Step inside the snmp directory and create a source/java directory
- Create a a new maven.xml file that looks as following:

```
<project default="jar:install"
        xmlns:j="jelly:core"
        xmlns:maven="jelly:maven"
        xmlns:ant="jelly:ant">

        <goal name='distro'>
   <attainGoal name='clean'/>
   <attainGoal name='jar'/>
   <delete dir='./dist' />
   <mkdir dir='./dist' />
   <mkdir dir='./dist/lib' />
   <copy todir="./dist/lib" file="${maven.build.dir}/${maven.final.name}.jar"/>
   <j:forEach var="lib" items="${pom.artifacts}">
      <j:set var="dep" value="${lib.dependency}"/>
      <j:if test="${dep.getProperty('war.bundle')=='true'}">
         <copy todir="./dist/lib" file="${lib.path}"/>
      </j:if>
   </j:forEach>
  </goal>

  <goal name='allBuild'>
  <attainGoal name='distro'/>
  </goal>
</project>
```

- Create a project.xml that looks as following:

```
<project>
  <pomVersion>3</pomVersion>
  <groupId>groundwork</groupId>
  <id>collage-adapter-snmp</id>
  <name>Groundwork Collage Adapters for SNMP</name>
  <currentVersion>1.1</currentVersion>

  <package>com.groundwork.feeder</package>

  <dependencies>
      <dependency>
      <id>collage-common-impl</id>
      <groupId>groundwork</groupId>
      <version>1.1</version>
    </dependency>
     <dependency>
      <id>collage-api</id>
      <groupId>groundwork</groupId>
      <version>1.1</version>
    </dependency>
     <dependency>
      <id>collage-adapter-api</id>
      <groupId>groundwork</groupId>
      <version>1.1</version>
    </dependency>

      <dependency>
      <id>collage-admin-impl</id>
      <groupId>groundwork</groupId>
      <version>1.1</version>
    </dependency>


    <dependency>
      <id>log4j</id>
      <version>1.2.8</version>
    </dependency>
    <dependency>
      <id>commons-logging</id>
      <version>1.0.3</version>
    </dependency>

     <dependency>
        <id>dom4j</id>
        <version>1.4-dev-8</version>
     </dependency>
    <dependency>
      <id>commons-collections</id>
      <version>3.0</version>
    </dependency>
    <dependency>
      <id>commons-lang</id>
      <version>2.0</version>
    </dependency>
  </dependencies>
   <build>
    <sourceDirectory>src/java</sourceDirectory>
     <resources>
      <resource>
        <directory>${basedir}/src/java</directory>
        <excludes>
          <exclude>**/*.java</exclude>
        </excludes>
      </resource>
    </resources>
  </build>
</project>
```

- Create a new package by creating the following directories under src/java: com.groundwork.feeder.adapter.impl

Now the setup of the new java project that will include the SNMP adapter is done. The next step will show what classes need to be implemented.

Classes and method to overwrite
The adapter has to implement the FeederBase interface which is part of the adapter-api.
Class creation:

- Create a new class SNMPTrap inside com/groundwork/feeder/adapter/impl that implements the FeederBase
- Implement GetName that returns the adapter name. The name has to match the node name of the XML fragment send to the listener. For snmp traps its SNMPTRAP
- Implement initialize() and uninitialize() for any actions that need to be executed when the adapter gets loaded or unloaded by the framework.
- Implement the method process() that gets called by the framework for each incoming xml stream the is of the adapter name (SNMPTRAP). Into this method goes the normalization code that transforms the XML message to a database call. The main steps are:
  - parse the XML stream and extarct the attributes. Use the utils.getAttributes() method
  - Get the API object by calling into the bean factory
  - Create a properties map and call into the API

Spring assemblies for SNMPTrap adapter
The new SNMPTrap adapter will be loaded into the Spring container. What you have to include into your JAR file is the spring assembly file which has to be created in the src/java/META-INF directory.
Steps:

- Create a new folder META-INF inside your project's snmp/src/java directory
- In META-INF create a file called assembly-adapter-snmptrap.xml. A sample of the whole file can be find in Appendix 2 Spring assembly for SNMPTrap adapter.

Building the adapter package
Inside the adapter/snmp directory execute:

```
maven jar
```

which will create the jar file in the target directory. Executing:

```
maven jar:install
```

will copy the jar file into the local repository.

Installing and configuring the SNMPTRAP adapter

Once the adapter is compiled successfully it needs to be deployed into the listener installation and the adapter.properties needs to be updated with the new adapter entry

- Copy the jar file (collage-adapter-snmp-1.1.jar) into the listener library path (/usr/local/groundwork/collage/feeder/lib)
- Edit adapter.properties for the new adapter as following:
  increment the counter for assemblies
  - `# Spring assemblies`
  - `nb.assemblies = 3`

  Add the assembly name and the Property Bean name
  - `# SNMPTrap Beans`
  - `adapter.assembly3 = META-INF/assembly-adapter-snmptrap.xml`
  - `adapter.properties.assembly3 = SNMPTrapAdapterProperties`
- Start the listener
- Feed data to the listener and verify if the data shows up in the database. Check the log files for any errors.

APPENDIX 1 SQL script for SNMPTRAP Metadata creation

```
# Database changes for SNMPTRAP messages

# Add new ApplicationType for SNMPTRAP

DELETE FROM ApplicationEntityProperty WHERE ApplicationTypeID = (SELECT ApplicationTypeID FROM ApplicationType WHERE
Name='SNMPTRAP') && EntityTypeID = (SELECT EntityTypeID FROM EntityType WHERE Name='LOG_MESSAGE');


REPLACE INTO ApplicationType (Name, Description) VALUES("SNMPTRAP","SNMP Trap application");

# Add the properties specific to SNMPTRAP

REPLACE INTO PropertyType(Name, Description, isString)  VALUES ("ipaddress", "ipdddress of snmp device", 1);
REPLACE INTO PropertyType(Name, Description, isString)  VALUES ("Event_OID_numeric", "Event_OID_numeric", 1);
REPLACE INTO PropertyType(Name, Description, isString)  VALUES ("Event_OID_symbolic", "Event_OID_symbolic of snmp device", 1);
REPLACE INTO PropertyType(Name, Description, isString)  VALUES ("Event_Name", "Event_Name", 1);
REPLACE INTO PropertyType(Name, Description, isString)  VALUES ("Category", "Category of snmp device", 1);
REPLACE INTO PropertyType(Name, Description, isString)  VALUES ("Variable_Bindings", "Variable_Bindings", 1);


# Assign the SNMP properties to Application Type SNMPTRAP and Entity LOG_MESSAGE

REPLACE INTO ApplicationEntityProperty(ApplicationTypeID, EntityTypeID, PropertyTypeID, SortOrder) VALUES ((SELECT ApplicationTypeID FROM
ApplicationType WHERE Name='SNMPTRAP'),(SELECT EntityTypeID FROM EntityType WHERE Name='LOG_MESSAGE'),(SELECT
PropertyTypeID FROM PropertyType WHERE Name = 'ipaddress'), 1);
REPLACE INTO ApplicationEntityProperty(ApplicationTypeID, EntityTypeID, PropertyTypeID, SortOrder) VALUES ((SELECT ApplicationTypeID FROM
ApplicationType WHERE Name='SNMPTRAP'),(SELECT EntityTypeID FROM EntityType WHERE Name='LOG_MESSAGE'),(SELECT
PropertyTypeID FROM PropertyType WHERE Name = 'Event_OID_numeric'), 1);
REPLACE INTO ApplicationEntityProperty(ApplicationTypeID, EntityTypeID, PropertyTypeID, SortOrder) VALUES ((SELECT ApplicationTypeID FROM
ApplicationType WHERE Name='SNMPTRAP'),(SELECT EntityTypeID FROM EntityType WHERE Name='LOG_MESSAGE'),(SELECT
PropertyTypeID FROM PropertyType WHERE Name = 'Event_OID_symbolic'), 1);
REPLACE INTO ApplicationEntityProperty(ApplicationTypeID, EntityTypeID, PropertyTypeID, SortOrder) VALUES ((SELECT ApplicationTypeID FROM
ApplicationType WHERE Name='SNMPTRAP'),(SELECT EntityTypeID FROM EntityType WHERE Name='LOG_MESSAGE'),(SELECT
PropertyTypeID FROM PropertyType WHERE Name = 'Event_Name'), 1);
REPLACE INTO ApplicationEntityProperty(ApplicationTypeID, EntityTypeID, PropertyTypeID, SortOrder) VALUES ((SELECT ApplicationTypeID FROM
ApplicationType WHERE Name='SNMPTRAP'),(SELECT EntityTypeID FROM EntityType WHERE Name='LOG_MESSAGE'),(SELECT
PropertyTypeID FROM PropertyType WHERE Name = 'Category'), 1);
REPLACE INTO ApplicationEntityProperty(ApplicationTypeID, EntityTypeID, PropertyTypeID, SortOrder) VALUES ((SELECT ApplicationTypeID FROM
ApplicationType WHERE Name='SNMPTRAP'),(SELECT EntityTypeID FROM EntityType WHERE Name='LOG_MESSAGE'),(SELECT
PropertyTypeID FROM PropertyType WHERE Name = 'Variable_Bindings'), 1);

#Create consolidation criteria

REPLACE INTO ConsolidationCriteria(Name, Criteria)  VALUES ('SNMPTRAP',
'Host;Severity;IpAddress;MonitorStatus;Event_OID_numeric;Event_Name;Category;Variable_Bindings');
```

APPENDIX 2 Spring Assembly for SNMPTrap adapter

The following file assembly-adapter-snmptrap.xml needs to be included into the jar package for the SNMPTrap adapter inside META-INF:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<!--
 List all the BeanID that have implemented the initialize method. The bean
Id's defined as a comma
 separated list will be called during the loading of the assembly
-->

<bean id="SNMPTrapAdapterProperties"
class="com.groundwork.feeder.adapter.impl.AdapterProperties">
      <constructor-arg
type="java.lang.String"><value>adapter.snmptrap</value></constructor-arg>
  </bean>

<bean id="adapter.snmptrap" singleton="false"
       class="com.groundwork.feeder.adapter.impl.SNMPTrap" />

</beans>
```