NetKarmaGUSHAdaptor Tool
User Manual
V2.5August 26, 2011

DATA TO INSIGHT CENTER

INDIANA UNIVERSITY

Pervasive Technology Institute

This document contains instructions for building and installing the Karma Adaptor Tool, v2.5, which provides core capability to derive, capture and store provenance events from experiment logs into the Karma Provenance repository and returns results. Karma Adaptor Tool is licensed under Apache License, Version 2.0 (the "License") (http://www.apache.org/licenses/LICENSE-2.0). The code is copyrighted and copyright owned by The Trustees of Indiana University. Adaptor tool is a product of the Data to Insight Center at Indiana University. See http://pti.iu.edu/d2i/provenance for more information.

Contents

1. Introduction

Karma Adaptor is one of the collection tools that make up the Karma provenance collection toolkit to harvest provenance from log files. It uses a rule file specific to an application to map raw data into Karma specific provenance events. The provenance of the data is stored into a relational database which can be visualized through various plugins. The provenance data can be used by the researcher to analyze their data, allow for the suspension and resumption of an experiment and provide references to find the details and data collected in an experiment.

This document describes 1) how to write a new rule file based on the rule set semantics for generating Karma specific provenance events, and 2) an application using the adaptor methodology for parsing application log files and ingesting provenance events to a Karma provenance repository.

2. Software Dependencies

Karma Adaptor v2.5 has been tested with the following software packages on which it has a dependency. These packages will need to be installed separately:

2.1 Installation Dependencies

1. Apache ANT v1.6 or higher (for building the tool from source)
   http://ant.apache.org
2. Java Development Kit (JDK) v5 or v6
   http://java.sun.com

2.2 Service Dependencies

Figure 1 shows how Karma Adaptor fits in the Karma Provenance Toolkit. Karma Adaptor requires the existence of two servers. The following are two servers, which are used for ingesting provenance data into a provenance repository:

1. Karma Service: Karma is a standalone tool that can be added to existing cyber-infrastructure for purposes of collection and representation of provenance data. The derived provenance events are sent to the Karma service through either an enterprise messaging bus (RabbitMQ) or a web service interface. The Karma Adaptor collection tool currently uses the RabbitMQ interface to ingest provenance events into the database for a highly reliable and scalable system. The URL for downloading Karma is:
   http://pti.iu.edu/d2i/provenance_karma
   If you have the Karma servicehosted on a server, there is no additional Karma dependency or download required to use the Karma Adaptor.
2. RabbitMQ: It is a messaging system, which the Adaptor tool uses to send messages to Karma. The RabbitMQ client is included in the Karma Adaptor, so as long as RabbitMQ is loaded on the server hosting the Karma service, there is no additional dependency for the adaptor itself.

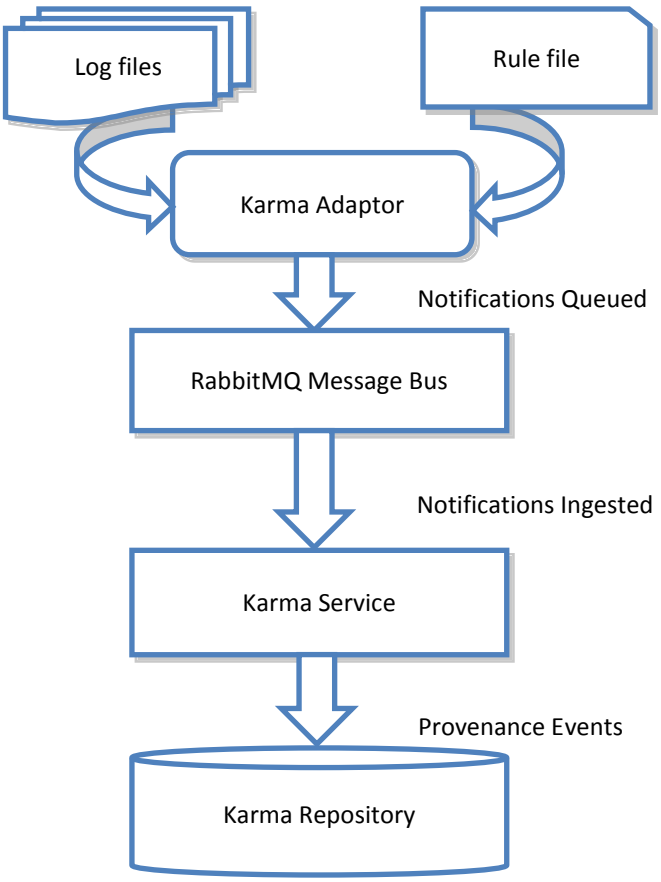

Figure 1. Karma Adaptor in Karma Provenance Toolkit

3. Configuring Adaptor Properties

Unzip the karma-adaptor-2.5.tar.gz, which contains the Karma-adaptor and provenance collection client to parse experiment logs and ingest provenance into the Karma repository, as:

```
tar xvzf karma-adaptor-2.5.tar.gz
```

This will create a directory named Karma-Adaptor which we refer to in the remainder of this manual as ${adaptor-home}. The adaptor depends on a number of properties for the correct execution. The derivation rules for converting raw data into Karma specific provenance events are described using a rule-file with a definite semantics. There is a default template rule-file present in /deps directory (ruleset.xml) which may need to be modified based on the target application. For gush logs, there are two generic rule-files, based on how an application is launched using gush. **For gush logs generated by using shell commands, cmdline_ruleset.xml is used whereas for experiments using application description xml, app_desc_ruleset.xml is used. In most cases, gush users don't have to specify their own rule-files and the adaptor will automatically select the one which suits the best.** Table-1 summarizes the classification.

In general, each application should have a specific rule file for parsing information from all log files generated by the application.

Table-1: Rule-file classification

Rule-file	Types of Log-files
app_desc_ruleset.xml	Gush logs generated using application description XML
cmdline_ruleset.xml	Gush logs generated using shell commands

The distribution package contains a properties file, **karma-adaptor.properties**, which can be found in the ${adaptor-home}/config directory. Please use this sample file to configure Karma according to your deployment environment. Below is a detailed explanation of the properties defined within this file:
The following 5 properties are used to connect to a RabbitMQ Server. If the instance of RabbitMQ already exists, contact the system admin for details.

messaging.username -- RabbitMQ Username
messaging.password -- RabbitMq Password
messaging.hostname -- Hostname that hosts RabbitMQ Server
messaging.hostport -- Port number on the RabbitMQ Server
messaging.virtualhost -- The virtual host is created for administrative purposes. Each connection (and all channels inside) must be associated with a single virtual host. Each virtual host comprises of its own namespace, a set of exchanges, message queues and all associated objects. The default value is "/".

The following 3 properties are used to configure how to send the Notifications to Karma Server.

messaging.exchangename -- A message routing agent. It can be durable (our system uses durable), temporary, and auto-deleted. Each message is delivered to each qualifying queue. The default value here is "KarmaExchange".

messaging.queuename -- Named "Weak FIFO" buffer. The default value is "KarmaQueue".
messaging.routingkey -- In our implementation, we use direct exchange type. Same routingkey is used on both publisher and subscriber sides. The default value here is "KarmaKey"

The following 2 properties are used to select the default rule-files for gush. These 2 properties should NOT be changed unless required.

karma-adaptor.experiment_rulefile -- ruleset file defining the rules for experiment log files.
karma-adaptor.cmdline_rulefile -- defines the rules for log files using gush shell-commands.

4. Writing a new rule-file

The rule file is a mapping document which specifies a set of rules to convert raw textual information present in a log file into a set of provenance events as managed by Karma. The rule file is an XML file

defined by a rule-set schema (snippet below) to identify and map the raw data. The full rule-set schema can be found in Appendix-B.

```
<elementname="karmaNotifications">
<complexType>
<sequence>
<elementname="project"type="string"/>
<elementname="argDelimiter"type="string"/>
<elementname="maxArgs"type="int"minOccurs="0"/>
<elementname="instanceID"type="Adaptor:instanceType"minOccurs="0"/>
<elementname="startTime"type="Adaptor:timestampType"minOccurs="0"/>
<elementname="ruleset"type="Adaptor:rule"minOccurs="1"
maxOccurs="unbounded"/>
<elementname="dependencyData"type="Adaptor:dependencyDataType"
minOccurs="0"maxOccurs="unbounded"/>
<elementname="dependency"type="Adaptor:dependencyType"
minOccurs="0"maxOccurs="unbounded"/>
<elementname="dependencyLink"type="Adaptor:dependencyLinkType"
minOccurs="0"maxOccurs="unbounded"/>
<elementname="globalDependency"type="Adaptor:dependencyType"
minOccurs="0"maxOccurs="unbounded"/>
</sequence>
</complexType>
</element>
```

The main elements of mapping the raw data into provenance events are defined under the karmaNotifications element as showed above. Each **ruleset** element (inside karmaNotifications element) in the rule file parses log lines to derive provenance events for an instance of a workflow and in terms of OPM defines process(es), artifact(s), agent(s), annotation(s) or a combination of any or all. The rule file contains of three major parts:

i) Rules for parsing the log files and associating each instance of a workflow execution to a log file. The XML elements in the rule file are: project, *argDelimiter*, *maxArgs*, *instanceID*, *startTime*.

ii) Rules for mapping textual data into Karma provenance events. The XML element is: *ruleset*.

iii) Rules for manipulating provenance events to enrich information representation. These elements are: *dependencyData*, *dependencyLink*, *dependency*, and *globalDependency*.

Each element is described in detail below:

[1] *project*: Name of the project. An example is a name of the workflow-engine. See sample below.

```
<netkarma:project>gush</netkarma:project>
```

[2] *argDelimiter*: Delimiter to separate arguments for parsing. An example of argument delimiter is shown below.

```
<netkarma:argDelimiter>:</netkarma:argDelimiter>
```

[3] *maxArgs*: Number of arguments each line of the log file should be parsed into. See sample below.

```
<netkarma:maxArgs>4</netkarma:maxArgs>
```

[4] *instanceID*: Contains rule to assign unique instance-ids for each workflow execution. See sample below.

```
<netkarma:instanceID>
<netkarma:instanceLocator>FILE</netkarma:instanceLocator>
<netkarma:fileInstanceLocator>
<netkarma:selectionMechanism>SIMPLE</netkarma:selectionMechanism>
<netkarma:simpleSelection>
<netkarma:selectionType>SUBSTRING</netkarma:selectionType>
<netkarma:substring>
<netkarma:argumentNumber>-1</netkarma:argumentNumber>
<netkarma:beginIndex>8</netkarma:beginIndex>
</netkarma:substring>
</netkarma:simpleSelection>
</netkarma:fileInstanceLocator>
</netkarma:instanceID>
```

[5] *startTime*: Defines rule to obtain the start-time, if available. See sample below.

```
<netkarma:startTime>
<netkarma:timestampLocator>FILE</netkarma:timestampLocator>
<netkarma:fileTimestampLocator>
<netkarma:selectionMechanism>SIMPLE</netkarma:selectionMechanism>
<netkarma:simpleSelection>
<netkarma:selectionType>LAST_NCHAR</netkarma:selectionType>
<netkarma:lastNChar>
<netkarma:argumentNumber>-1</netkarma:argumentNumber>
<netkarma:numChars>10</netkarma:numChars>
</netkarma:lastNChar>
</netkarma:simpleSelection>
</netkarma:fileTimestampLocator>
</netkarma:startTime>
```

[6] *ruleset*: Set of rules to map raw data into provenance events compatible with Karma. Each of these rulesets corresponds to deriving an entity (process/artifact/annotation/agent) in OPM and a part or full notification in Karma. See sample ruleset below.

```
<!-- data-block part of a Workflow-Invoked notification in Karma -->
<netkarma:ruleset>
<netkarma:hasDuplicates>true</netkarma:hasDuplicates>
<netkarma:filter>
<netkarma:argumentNumber>0</netkarma:argumentNumber>
<netkarma:argumentValue>gush.cc</netkarma:argumentValue>
<netkarma:comparator>EQUALS</netkarma:comparator>
</netkarma:filter>
<netkarma:filter>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:argumentValue>Gush constructor: port</netkarma:argumentValue>
<netkarma:filterPredicate>AND</netkarma:filterPredicate>
<netkarma:comparator>CONTAINS</netkarma:comparator>
</netkarma:filter>
<netkarma:notification>
<netkarma:notificatonId>1</netkarma:notificatonId>
```

```
<netkarma:notificationType>WORKFLOW_INVOKED</netkarma:notificationType>
<netkarma:notificationPartType>DATA_BLOCK</netkarma:notificationPartType>
<netkarma:dataBlocks>
<netkarma:dataId>config-param</netkarma:dataId>
<netkarma:dataType>BLOCK</netkarma:dataType>
<netkarma:dataValue>
<netkarma:uriInfo>
<netkarma:identifier>port</netkarma:identifier>
<netkarma:type>URN</netkarma:type>
</netkarma:uriInfo>
<netkarma:selectMethod>
<netkarma:selectionType>SUBSTRING</netkarma:selectionType>
<netkarma:substring>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:beginIndex>24</netkarma:beginIndex>
</netkarma:substring>
</netkarma:selectMethod>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
</netkarma:dataValue>
</netkarma:dataBlocks>
<netkarma:notificationTime>
<netkarma:timestampLocator>DERIVED</netkarma:timestampLocator>
<netkarma:timestamp>
<netkarma:selectionType>COMPLETE_STRING</netkarma:selectionType>
<netkarma:completeString>
<netkarma:argumentNumber>2</netkarma:argumentNumber>
</netkarma:completeString>
</netkarma:timestamp>
</netkarma:notificationTime>
</netkarma:notification>
</netkarma:ruleset>
```

[7] *dependencyData*: Optional rules to identify certain data which might be dependent on some other information in the log file to redefine entities and relationships of initially generated events.

```
<netkarma:dependencyData>
<netkarma:name>HOST</netkarma:name>
<netkarma:filter>
<netkarma:argumentNumber>0</netkarma:argumentNumber>
<netkarma:argumentValue>process_block.cc</netkarma:argumentValue>
<netkarma:comparator>EQUALS</netkarma:comparator>
</netkarma:filter>
<netkarma:filter>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:argumentValue>Client has notified</netkarma:argumentValue>
<netkarma:filterPredicate>AND</netkarma:filterPredicate>
<netkarma:comparator>CONTAINS</netkarma:comparator>
</netkarma:filter>
<netkarma:value>
<netkarma:selectionMechanism>COMPLEX</netkarma:selectionMechanism>
<netkarma:complexSelection>
<netkarma:simpleSelection>
<netkarma:selectionType>READ_TOKEN</netkarma:selectionType>
<netkarma:readToken>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
```

```
<netkarma:delimiter>@</netkarma:delimiter>
<netkarma:maxTokens>2</netkarma:maxTokens>
<netkarma:tokenNumbers>2</netkarma:tokenNumbers>
</netkarma:readToken>
</netkarma:simpleSelection>
</netkarma:complexSelection>
</netkarma:value>
</netkarma:dependencyData>
```

[8] *dependency*: Rules to define matching rules in order to redefine events with modified data. This element defines any complex rule to modify the name of a process, which derived by parsing a line in the log file for mapping the data into a notification, into a value which derived from some other notification for the same process. A sample dependency rule is shown below.

```
<!-- Dependency: process -> experiment-output -->
<netkarma:dependency>
<netkarma:sourceNotificationId>7</netkarma:sourceNotificationId>
<netkarma:targetNotificationId>3</netkarma:targetNotificationId>
<netkarma:sourceActorType>INVOKEE</netkarma:sourceActorType>
<netkarma:targetActorType>PRODUCER</netkarma:targetActorType>
<netkarma:matchRule>
<netkarma:matchLineType>PREV_NTH_LINE</netkarma:matchLineType>
<netkarma:matchLineNum>1</netkarma:matchLineNum>
<netkarma:matchDataName>PID</netkarma:matchDataName>
<netkarma:matchDataValue>
<netkarma:selectionMechanism>SIMPLE</netkarma:selectionMechanism>
<netkarma:simpleSelection>
<netkarma:selectionType>READ_TOKEN</netkarma:selectionType>
<netkarma:readToken>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:delimiter>=</netkarma:delimiter>
<netkarma:maxTokens>2</netkarma:maxTokens>
<netkarma:tokenNumbers>2</netkarma:tokenNumbers>
</netkarma:readToken>
</netkarma:simpleSelection>
</netkarma:matchDataValue>
</netkarma:matchRule>
</netkarma:dependency>
```

[9] *dependencyLink*: Creates links between defined dependency-data elements. See sample below.

```
<netkarma:dependencyLink>
<netkarma:linkType>SEQUENTIAL</netkarma:linkType>
<netkarma:source>PID</netkarma:source>
<netkarma:target>PROCESS</netkarma:target>
</netkarma:dependencyLink>
```

[10] *globalDependency*: Optional rules to redefine the overall structure of the provenance graph. See sample below.

```
<netkarma:globalDependency>
<netkarma:sourceNotificationId>4</netkarma:sourceNotificationId>
<netkarma:targetNotificationId>7</netkarma:targetNotificationId>
<netkarma:sourceActorType>INVOKEE</netkarma:sourceActorType>
<netkarma:targetActorType>INVOKER</netkarma:targetActorType>
```

```
</netkarma:globalDependency>
```

A simplified rule file is described in Appendix-C. A detailed sample rule-file (ruleset.xml) and the XML schema (ruleset.xsd) defining the rule semantics are present in the deps/ directory.

5. Building and Executing the Adaptor

This section describes how to build and use the Adaptor code when Karma Server is hosted as standalone server using the RabbitMQ messaging system.

5.1 Build

Both <JAVA_HOME> and <ANT_HOME> environment variables should be set before building the Adaptor code using ANT:

```
ant karma-adaptor
```

5.2 Execution

The environment parameters for executing the script to capture provenance should be set in the configuration-file, **adaptor_stdenvs.cfg**, located in the ${adaptor-home} directory.

```
vi adaptor_stdenvs.cfg
```

The <ADAPTOR_HOME> and <JAVA_HOME> path have to be set in the configuration file.

```
ADAPTOR_HOME= <absolute path to the ${adaptor-home} directory the adaptor-code is extracted to>
JAVA_HOME= <put your Java home here>
```

The script to parse log files and ingest provenance events into the Karma repository is executed as follows from within the ${adaptor-home} directory:

```
./provenance_collector.sh -l <path to logfile>
```

where <logfile> is the name of the log file to be used for provenance collection.

If the user wants to override the existing rule-files with their custom rule-files, the script should be executed as:

```
./provenance_collector.sh -l <path to logfile> -r < path to rulefile>
```

where <rulefile> is the name of the custom rule-file to be used.

Sample execution

```
./provenance -l logfile-karma-virtual-machine-15556-1288050231.txt
Connecting to Server...
Using rulefile: /home/workspace/Karma-Adaptor/deps/app_desc_ruleset.xml
Creating Notifications...
Applying dependency rules...
Ingesting Notifications...
Number of Notifications = 56
Workflow Instance-ID: urn:tool:gush:karma-virtual-machine-15556-1288050231e6d0c01b-d0f4-4cc6-91ab-9d6a1b6d572e
Time to queue notifications for ingestion: 19.814 secs
```

The Workflow Instance-ID printed in the standard-output is the unique identifier to extract complete provenance graphs from the Karma repository using the query clients and visualization tools.

6. Visualization

Provenance retrieval and visualization plugins for Cytoscape can be downloaded from the following website to visualize provenance graphs:
http://pti.iu.edu/d2i/provenance_karma

Appendix-A: Rulefile Skeleton

Described below is a basic skeleton of a rule-file. This skeleton shows the mandatory elements required for creating a rulefile as per the descriptions above. The values in square-brackets have to be substituted either with application-specific constants or with element-type definitions defined in the XML schema described above.

```
<netkarma:project>[project-name]</netkarma:project>
<netkarma:argDelimiter>[argument-delimiter]</netkarma:argDelimiter>
<netkarma:ruleset>
<netkarma:hasDuplicates>[true/false]</netkarma:hasDuplicates>
<netkarma:filter>
<netkarma:argumentNumber>[filter-argnum]</netkarma:argumentNumber>
<netkarma:argumentValue>[ value-to-compare]</netkarma:argumentValue>
<netkarma:comparator>[EQUALS/CONTAINS/…]</netkarma:comparator>
</netkarma:filter>
<netkarma:notification>
<netkarma:notificatonId>[unique-notification-id]</netkarma:notificatonId>
<netkarma:notificationType>[notification-type]</netkarma:notificationType>
<netkarma:notificationPartType>[subtype]</netkarma:notificationPartType>
<netkarma:[subtype]>
[subtype-type]
</netkarma:[subtype]>
<netkarma:notificationTime>
<netkarma:timestampLocator>[derivation-type]</netkarma:timestampLocator>
<netkarma:timestamp>
                    [timestamp-type]
</netkarma:timestamp>
</netkarma:notificationTime>
</netkarma:notification>
</netkarma:ruleset>
```

Appendix-B: RulesetSchema

```
<?xmlversion="1.0"encoding="UTF-8"?>
<schemaelementFormDefault="qualified"
targetNamespace="http://www.dataandsearch.org/netkarma/"
xmlns:netkarma="http://www.dataandsearch.org/netkarma/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema">

<!-- ================ -->
<!-- TYPE DEFINITIONS -->
<!-- ================ -->

<simpleTypename="notificationEnumType">
<restrictionbase="xsd:string">
<enumerationvalue="INVOKING_SERVICE"/>
<enumerationvalue="SERVICE_INVOKED"/>
<enumerationvalue="INVOKING_WORKFLOW"/>
<enumerationvalue="WORKFLOW_INVOKED"/>
<enumerationvalue="DATA_PRODUCED"/>
<enumerationvalue="DATA_CONSUMED"/>
<enumerationvalue="DATA_SEND_STARTED"/>
<enumerationvalue="DATA_RECEIVE_STARTED"/>
```

```
<enumeration value="DATA_SEND_FINISHED"/>
<enumeration value="DATA_RECEIVE_FINISHED"/>
</restriction>
</simpleType>

<simpleType name="actorEnumType">
<restriction base="string">
<enumeration value="INVOKER"/>
<enumeration value="INVOKEE"/>
<enumeration value="PRODUCER"/>
<enumeration value="CONSUMER"/>
<enumeration value="SENDER"/>
<enumeration value="RECEIVER"/>
</restriction>
</simpleType>

<simpleType name="dataEnumType">
<restriction base="string">
<enumeration value="FILE"/>
<enumeration value="BLOCK"/>
</restriction>
</simpleType>

<simpleType name="timestampLocatorType">
<restriction base="string">
<enumeration value="DERIVED"/>
<enumeration value="FILE"/>
<enumeration value="HEADER"/>
<enumeration value="FOOTER"/>
</restriction>
</simpleType>

<simpleType name="instanceLocatorType">
<restriction base="string">
<enumeration value="DERIVED"/>
<enumeration value="FILE"/>
<enumeration value="HEADER"/>
</restriction>
</simpleType>

<simpleType name="actorEntityEnumType">
<restriction base="string">
<enumeration value="USER"/>
<enumeration value="WORKFLOW"/>
<enumeration value="SERVICE"/>
<enumeration value="METHOD"/>
</restriction>
</simpleType>

<simpleType name="parentEnumType">
<restriction base="string">
<enumeration value="USER"/>
<enumeration value="WORKFLOW"/>
<enumeration value="SERVICE"/>
</restriction>
</simpleType>

<simpleType name="actorEntityEnumSubtype">
<restriction base="string">
<enumeration value="CONTROLLER"/>
```

```
<enumeration value="HUMAN_PROXY"/>
</restriction>
</simpleType>

<simpleType name="comparatorEnumType">
<restriction base="string">
<enumeration value="EQUALS"/>
<enumeration value="CONTAINS"/>
</restriction>
</simpleType>

<simpleType name="notificationPartEnumType">
<restriction base="string">
<enumeration value="ACTOR"/>
<enumeration value="DATA_BLOCK"/>
<enumeration value="ANNOTATION"/>
<enumeration value="ALL"/>
</restriction>
</simpleType>

<simpleType name="uriEnumType">
<restriction base="string">
<enumeration value="URL"/>
<enumeration value="URN"/>
</restriction>
</simpleType>

<simpleType name="filterPredicateEnumType">
<restriction base="string">
<enumeration value="AND"/>
<enumeration value="OR"/>
</restriction>
</simpleType>

<simpleType name="selectMethodEnumType">
<restriction base="string">
<enumeration value="COMPLETE_STRING"/>
<enumeration value="SUBSTRING"/>
<enumeration value="READ_PROPERTY_OF"/>
<enumeration value="READ_ARGUMENT_OF"/>
<enumeration value="READ_TOKEN"/>
<enumeration value="CONSTANT"/>
<enumeration value="LAST_NCHAR"/>
</restriction>
</simpleType>

<simpleType name="selectMechanismEnumType">
<restriction base="string">
<enumeration value="SIMPLE"/>
<enumeration value="COMPLEX"/>
</restriction>
</simpleType>

<simpleType name="autoIncrementEnumType">
<restriction base="string">
<enumeration value="AUTO_INCREMENT"/>
</restriction>
</simpleType>

<simpleType name="matchLineEnumType">
```

```
<restrictionbase="string">
<enumerationvalue="PREV_NTH_LINE"/>
<enumerationvalue="NEXT_NTH_LINE"/>
</restriction>
</simpleType>

<simpleTypename="dependencyLinkEnumType">
<restrictionbase="string">
<enumerationvalue="SEQUENTIAL"/>
<enumerationvalue="DIRECT"/>
</restriction>
</simpleType>

<complexTypename="appenderType">
<choice>
<elementname="autoIncrement"type="netkarma:autoIncrementEnumType"/>
<elementname="selectMethod"type="netkarma:selectMethodType"/>
</choice>
</complexType>

<complexTypename="timestampType">
<sequence>
<elementname="timestampLocator"type="netkarma:timestampLocatorType"/>
<choice>
<elementname="timestamp"type="netkarma:selectMethodType"/>
<elementname="fileTimestampLocator"type="netkarma:selectMethodType"/>
<elementname="headerTimestampLocator"type="netkarma:selectMethodType"/>
<elementname="footerTimestampLocator"type="netkarma:selectMethodType"/>
</choice>
</sequence>
</complexType>

<complexTypename="instanceType">
<sequence>
<elementname="instanceLocator"type="netkarma:instanceLocatorType"/>
<choice>
<elementname="derivedInstanceLocator"type="netkarma:provenanceDataType"/>
<elementname="fileInstanceLocator"type="netkarma:selectMethodType"/>
<elementname="headerInstanceLocator"type="netkarma:selectMethodType"/>
</choice>
</sequence>
</complexType>

<complexTypename="dependencyType">
<sequence>
<elementname="sourceNotificationId"type="int"/>
<elementname="targetNotificationId"type="int"/>
<elementname="sourceActorType"type="netkarma:actorEnumType"/>
<elementname="targetActorType"type="netkarma:actorEnumType"/>
<elementname="matchRule"type="netkarma:matchRuleType"minOccurs="0"maxOccurs="unbou
nded"/>
</sequence>
</complexType>

<complexTypename="matchRuleType">
<sequence>
<elementname="matchLineType"type="netkarma:matchLineEnumType"/>
<elementname="matchLineNum"type="int"/>
<elementname="matchDataName"type="string"/>
<elementname="matchDataValue"type="netkarma:selectMethodType"/>
```

```
</sequence>
</complexType>

<complexTypename="dependencyDataType">
<sequence>
<elementname="name"type="string"/>
<elementname="filter"type="netkarma:filterType"minOccurs="1"maxOccurs="unbounded"/
>
<elementname="value"type="netkarma:selectMethodType"/>
</sequence>
</complexType>

<complexTypename="dependencyLinkType">
<sequence>
<elementname="linkType"type="netkarma:dependencyLinkEnumType"/>
<elementname="source"type="string"/>
<elementname="target"type="string"/>
</sequence>
</complexType>

<complexTypename="simpleSelectionType">
<sequence>
<elementname="selectionType"type="netkarma:selectMethodEnumType"/>
<choice>
<elementname="completeString"type="netkarma:completeType"/>
<elementname="substring"type="netkarma:substringType"/>
<elementname="readPropertyOf"type="netkarma:readPropertyOfType"/>
<elementname="readArgumentOf"type="netkarma:readArgumentOfType"/>
<elementname="readToken"type="netkarma:readTokenType"/>
<elementname="constant"type="netkarma:constantType"/>
<elementname="lastNChar"type="netkarma:lastNCharType"/>
</choice>
</sequence>
</complexType>

<complexTypename="complexSelectionType">
<sequence>
<elementname="simpleSelection"type="netkarma:simpleSelectionType"minOccurs="2"maxO
ccurs="unbounded"/>
</sequence>
</complexType>

<complexTypename="selectMethodType">
<sequence>
<elementname="selectionMechanism"type="netkarma:selectMechanismEnumType"/>
<choice>
<elementname="simpleSelection"type="netkarma:simpleSelectionType"/>
<elementname="complexSelection"type="netkarma:complexSelectionType"/>
</choice>
</sequence>
</complexType>

<complexTypename="lastNCharType">
<sequence>
<elementname="argumentNumber"type="int"/>
<elementname="numChars"type="int"/>
</sequence>
</complexType>

<complexTypename="completeType">
```

```
<sequence>
<elementname="argumentNumber"type="int"/>
</sequence>
</complexType>

<complexTypename="constantType">
<sequence>
<elementname="constantValue"type="string"/>
</sequence>
</complexType>

<complexTypename="substringType">
<sequence>
<elementname="argumentNumber"type="int"/>
<elementname="beginIndex"type="int"/>
<elementname="endIndex"type="int"minOccurs="0"/>
</sequence>
</complexType>

<complexTypename="readPropertyOfType">
<sequence>
<elementname="argumentNumber"type="int"/>
<elementname="key"type="string"/>
<elementname="delimiter"type="string"minOccurs="0"/>
</sequence>
</complexType>

<complexTypename="readArgumentOfType">
<sequence>
<elementname="argumentNumber"type="int"/>
<elementname="methodName"type="string"/>
<elementname="paramDelimiter"type="string"minOccurs="0"/>
<elementname="paramList"type="int"minOccurs="1"maxOccurs="unbounded"/>
</sequence>
</complexType>

<complexTypename="readTokenType">
<sequence>
<elementname="argumentNumber"type="int"/>
<elementname="delimiter"type="string"/>
<elementname="maxTokens"type="int"/>
<elementname="tokenNumbers"type="int"minOccurs="1"maxOccurs="unbounded"/>
</sequence>
</complexType>

<!-- ================= -->
<!-- MAJOR DEFINITIONS -->
<!-- ================= -->

<complexTypename="actorsType">
<sequence>
<elementname="actorId"type="netkarma:provenanceDataType"/>
<elementname="appendId"type="netkarma:appenderType"minOccurs="0"maxOccurs="unbound
ed"/>
<elementname="actorType"type="netkarma:actorEnumType"/>
<elementname="entityType"type="netkarma:actorEntityEnumType"/>
<elementname="onBehalfOf"type="netkarma:provenanceDataType"minOccurs="0"/>
<elementname="entitySubtype"type="netkarma:actorEntityEnumSubtype"minOccurs="0"/>
<elementname="annotations"type="netkarma:annotationType"minOccurs="0"maxOccurs="un
bounded"/>
```

```
<elementname="parent"type="netkarma:parentType"minOccurs="0"maxOccurs="unbounded"/
>
<elementname="timestep"type="xsd:int"minOccurs="0"/>
</sequence>
</complexType>

<complexTypename="dataType">
<sequence>
<elementname="dataId"type="xsd:string"/>
<elementname="appendId"type="netkarma:appenderType"minOccurs="0"maxOccurs="unbound
ed"/>
<elementname="dataType"type="netkarma:dataEnumType"/>
<elementname="dataValue"type="netkarma:provenanceDataType"/>
<elementname="annotations"type="netkarma:annotationType"minOccurs="0"maxOccurs="un
bounded"/>
</sequence>
</complexType>

<complexTypename="annotationType">
<sequence>
<elementname="property"type="xsd:string"/>
<elementname="value"type="netkarma:provenanceDataType"/>
</sequence>
</complexType>

<complexTypename="filterType">
<sequence>
<elementname="argumentNumber"type="int"/>
<elementname="argumentValue"type="string"/>
<elementname="filterPredicate"type="netkarma:filterPredicateEnumType"minOccurs="0"
/>
<elementname="comparator"type="netkarma:comparatorEnumType"/>
</sequence>
</complexType>

<complexTypename="provenanceDataType">
<sequence>
<elementname="uriInfo"type="netkarma:uriType"/>
<elementname="selectMethod"type="netkarma:selectMethodType"/>
<elementname="argumentNumber"type="int"/>
</sequence>
</complexType>

<complexTypename="parentType">
<sequence>
<elementname="parentId"type="netkarma:provenanceDataType"/>
<elementname="parentType"type="netkarma:parentEnumType"/>
</sequence>
</complexType>

<complexTypename="uriType">
<sequence>
<elementname="identifier"type="xsd:string"/>
<elementname="type"type="netkarma:uriEnumType"/>
</sequence>
</complexType>

<!-- karma-notification element -->

<complexTypename="notificationType">
```

```
<sequence>
<elementname="notificatonId"type="xsd:int"/>
<elementname="notificationType"type="netkarma:notificationEnumType"/>
<elementname="notificationPartType"type="netkarma:notificationPartEnumType"/>
<elementname="actors"type="netkarma:actorsType"minOccurs="0"maxOccurs="unbounded"/
>
<elementname="dataBlocks"type="netkarma:dataType"minOccurs="0"maxOccurs="unbounded
"/>
<elementname="annotations"type="netkarma:annotationType"minOccurs="0"maxOccurs="un
bounded"/>
<elementname="notificationTime"type="netkarma:timestampType"minOccurs="0"/>
</sequence>
</complexType>

<complexTypename="rule">
<sequence>
<elementname="hasDuplicates"type="boolean"/>
<elementname="isDistinct"type="boolean"minOccurs="0"/>
<elementname="filter"type="netkarma:filterType"minOccurs="1"maxOccurs="unbounded"/
>
<elementname="notification"type="netkarma:notificationType"/>
</sequence>
</complexType>

<elementname="karmaNotifications">
<complexType>
<sequence>
<elementname="project"type="string"/>
<elementname="argDelimiter"type="string"/>
<elementname="maxArgs"type="int"minOccurs="0"/>
<elementname="instanceID"type="netkarma:instanceType"minOccurs="0"/>
<elementname="startTime"type="netkarma:timestampType"minOccurs="0"/>
<elementname="ruleset"type="netkarma:rule"minOccurs="1"maxOccurs="unbounded"/>
<elementname="dependencyData"type="netkarma:dependencyDataType"minOccurs="0"maxOcc
urs="unbounded"/>
<elementname="dependency"type="netkarma:dependencyType"minOccurs="0"maxOccurs="unb
ounded"/>
<elementname="dependencyLink"type="netkarma:dependencyLinkType"minOccurs="0"maxOcc
urs="unbounded"/>
<elementname="globalDependency"type="netkarma:dependencyType"minOccurs="0"maxOccur
s="unbounded"/>
</sequence>
</complexType>
</element>
</schema>
```

Appendix-C: Sample Rule File.

```
<netkarma:karmaNotificationsxmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xmlns="http://www.dataandsearch.org/netkarma/"xmlns:netkarma="http://www.
dataandsearch.org/netkarma/"
xsi:schemaLocation="http://www.dataandsearch.org/netkarma/
deps/notification_ruleset.xsd">
<netkarma:project>gush</netkarma:project>
<netkarma:argDelimiter>:</netkarma:argDelimiter>
<netkarma:maxArgs>4</netkarma:maxArgs>
<!-- instanceID or WorkflowID -->
<netkarma:instanceID>
<netkarma:instanceLocator>FILE</netkarma:instanceLocator>
<netkarma:fileInstanceLocator>
<netkarma:selectionMechanism>SIMPLE</netkarma:selectionMechanism>
<netkarma:simpleSelection>
<netkarma:selectionType>SUBSTRING</netkarma:selectionType>
<netkarma:substring>
<netkarma:argumentNumber>-1</netkarma:argumentNumber>
<netkarma:beginIndex>8</netkarma:beginIndex>
</netkarma:substring>
</netkarma:simpleSelection>
</netkarma:fileInstanceLocator>
</netkarma:instanceID>
<netkarma:startTime>
<netkarma:timestampLocator>FILE</netkarma:timestampLocator>
<netkarma:fileTimestampLocator>
<netkarma:selectionMechanism>SIMPLE</netkarma:selectionMechanism>
<netkarma:simpleSelection>
<netkarma:selectionType>LAST_NCHAR</netkarma:selectionType>
<netkarma:lastNChar>
<netkarma:argumentNumber>-1</netkarma:argumentNumber>
<netkarma:numChars>10</netkarma:numChars>
</netkarma:lastNChar>
</netkarma:simpleSelection>
</netkarma:fileTimestampLocator>
</netkarma:startTime>
<!-- data-block part of a Workflow-Invoked notification in Karma -->
<netkarma:ruleset>
<netkarma:hasDuplicates>true</netkarma:hasDuplicates>
<netkarma:filter>
<netkarma:argumentNumber>0</netkarma:argumentNumber>
<netkarma:argumentValue>gush.cc</netkarma:argumentValue>
<netkarma:comparator>EQUALS</netkarma:comparator>
</netkarma:filter>
<netkarma:filter>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:argumentValue>Gush constructor: port</netkarma:argumentValue>
<netkarma:filterPredicate>AND</netkarma:filterPredicate>
<netkarma:comparator>CONTAINS</netkarma:comparator>
</netkarma:filter>
```

```
<netkarma:notification>
<netkarma:notificatonId>1</netkarma:notificatonId>
<netkarma:notificationType>WORKFLOW_INVOKED</netkarma:notificationType>
<netkarma:notificationPartType>DATA_BLOCK</netkarma:notificationPartType>
<netkarma:dataBlocks>
<netkarma:dataId>config-param</netkarma:dataId>
<netkarma:dataType>BLOCK</netkarma:dataType>
<netkarma:dataValue>
<netkarma:uriInfo>
<netkarma:identifier>port</netkarma:identifier>
<netkarma:type>URN</netkarma:type>
</netkarma:uriInfo>
<netkarma:selectMethod>
<netkarma:selectionType>SUBSTRING</netkarma:selectionType>
<netkarma:substring>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:beginIndex>24</netkarma:beginIndex>
</netkarma:substring>
</netkarma:selectMethod>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
</netkarma:dataValue>
</netkarma:dataBlocks>
<netkarma:notificationTime>
<netkarma:timestampLocator>DERIVED</netkarma:timestampLocator>
<netkarma:timestamp>
<netkarma:selectionType>COMPLETE_STRING</netkarma:selectionType>
<netkarma:completeString>
<netkarma:argumentNumber>2</netkarma:argumentNumber>
</netkarma:completeString>
</netkarma:timestamp>
</netkarma:notificationTime>
</netkarma:notification>
</netkarma:ruleset>
<!-- Dependency-data: HOST -->
<netkarma:dependencyData>
<netkarma:name>HOST</netkarma:name>
<netkarma:filter>
<netkarma:argumentNumber>0</netkarma:argumentNumber>
<netkarma:argumentValue>process_block.cc</netkarma:argumentValue>
<netkarma:comparator>EQUALS</netkarma:comparator>
</netkarma:filter>
<netkarma:filter>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:argumentValue>Client has notified</netkarma:argumentValue>
<netkarma:filterPredicate>AND</netkarma:filterPredicate>
<netkarma:comparator>CONTAINS</netkarma:comparator>
</netkarma:filter>
<netkarma:value>
<netkarma:selectionMechanism>COMPLEX</netkarma:selectionMechanism>
<netkarma:complexSelection>
<netkarma:simpleSelection>
<netkarma:selectionType>READ_TOKEN</netkarma:selectionType>
```

```
<netkarma:readToken>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:delimiter>@</netkarma:delimiter>
<netkarma:maxTokens>2</netkarma:maxTokens>
<netkarma:tokenNumbers>2</netkarma:tokenNumbers>
</netkarma:readToken>
</netkarma:simpleSelection>
</netkarma:complexSelection>
</netkarma:value>
</netkarma:dependencyData>
<!-- Dependency: process -> experiment-output -->
<netkarma:dependency>
<netkarma:sourceNotificationId>7</netkarma:sourceNotificationId>
<netkarma:targetNotificationId>3</netkarma:targetNotificationId>
<netkarma:sourceActorType>INVOKEE</netkarma:sourceActorType>
<netkarma:targetActorType>PRODUCER</netkarma:targetActorType>
<netkarma:matchRule>
<netkarma:matchLineType>PREV_NTH_LINE</netkarma:matchLineType>
<netkarma:matchLineNum>1</netkarma:matchLineNum>
<netkarma:matchDataName>PID</netkarma:matchDataName>
<netkarma:matchDataValue>
<netkarma:selectionMechanism>SIMPLE</netkarma:selectionMechanism>
<netkarma:simpleSelection>
<netkarma:selectionType>READ_TOKEN</netkarma:selectionType>
<netkarma:readToken>
<netkarma:argumentNumber>3</netkarma:argumentNumber>
<netkarma:delimiter>=</netkarma:delimiter>
<netkarma:maxTokens>2</netkarma:maxTokens>
<netkarma:tokenNumbers>2</netkarma:tokenNumbers>
</netkarma:readToken>
</netkarma:simpleSelection>
</netkarma:matchDataValue>
</netkarma:matchRule>
</netkarma:dependency>
<!-- Dependency-link: pid->process -->
<netkarma:dependencyLink>
<netkarma:linkType>SEQUENTIAL</netkarma:linkType>
<netkarma:source>PID</netkarma:source>
<netkarma:target>PROCESS</netkarma:target>
</netkarma:dependencyLink>
<!-- Global Dependencies -->
<netkarma:globalDependency>
<netkarma:sourceNotificationId>4</netkarma:sourceNotificationId>
<netkarma:targetNotificationId>7</netkarma:targetNotificationId>
<netkarma:sourceActorType>INVOKEE</netkarma:sourceActorType>
<netkarma:targetActorType>INVOKER</netkarma:targetActorType>
</netkarma:globalDependency>
</netkarma:karmaNotifications>
```

Appendix-D: Sample Log File

```
serviceInvoked,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
cron,1,http://d2i.org/amsreprovenance/iu/filename-
daily,20101117181942,invoked,cron1,is the main orchestrator of the workflow for
sea ice processing,none::20091002::FINAL#none#none#Sea Ice
dataProduced,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
daily,1,http://d2i.org/amsreprovenance/iu/filename-
clientServiceID,1203084381,dataProduced4,type,empty-file produced for santa
processing,/var/tmp/daily-17618.std==14::none::none
serviceInvoked,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
daily,1,http://d2i.org/amsreprovenance/iu/filename-
santa,20101117181944,invoked,Daily1,Invocation to santa to find all the input
brightness tempraturefiles,none::none::none
dataConsumed,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
santa,1,http://d2i.org/amsreprovenance/iu/filename-
clientserviceID,20101117181944,dataConsumed,file6,makes a list of the the
brightness temperature files to be used,http://d2i.org/amsreprovenance/iu/envname-
env.pm==10::none::none
serviceInvoked,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
daily,1,http://d2i.org/amsreprovenance/iu/filename-
L3,20101117181944,invoked,Daily5,L33,/var/tmp/daily-17618.std==14::none::none
dataConsumed,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
L3,1,http://d2i.org/amsreprovenance/iu/filename-
clientServiceID,20101117181944,dataConsumed,file5,maskfile consumed by
L3,/ftp/ops/science/data/level3/seaice12/AMSR_E_L3_SeaIce12km_V12_20090930.hdf==12
::none::none
dataProduced,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
L3,1,http://d2i.org/amsreprovenance/iu/filename-
clientServiceID,20101117182916,dataProduced1,final-output produced by L3,the 12km
products,AMSR_E_L3_SeaIce12km_V12_20091002.hdf==15::none::none
dataProduced,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
L3,1,http://d2i.org/amsreprovenance/iu/filename-
clientServiceID,20101117182916,dataProduced2,final-output produced by L3,the 6km
products,AMSR_E_L3_SeaIce6km_V12_20091002.hdf==15::none::none
addAnnotations,sea-ice-processing-
20101117181942,level1,file,1,AMSR_E_L3_SeaIce6km_V12_20091002.hdf,1203084381,add-
annotation,Algorithm name,NT2,none::none::none
20101117181942,level1,file,1,AMSR_E_L3_SeaIce6km_V12_20091002.hdf,1203084381,add-
annotation,Algorithm name,SnowDepth,none::none::none
addAnnotations,sea-ice-processing-
20101117181942,level1,file,1,AMSR_E_L3_SeaIce6km_V12_20091002.hdf,1203084381,add-
annotation,Algorithm name,BBA,none::none::none
serviceInvoked,sea-ice-processing-
20101117181942,level1,http://d2i.org/amsreprovenance/iu/filename-
L3,1,http://d2i.org/amsreprovenance/iu/filename-
L3seaice,20101117181944,invoked,L32,executes the pge for producing the final sea
ice products,none::none::none
```