Apam History Manager   
HistMan

# Goal

Provide a simple way to get information about the past states of Apam. This specification is only a first and simple implementation used primarily to perform experiments and better understand the needs.

# Principle

Apam maintains a state (Apam State Model, or ASM) which is a graph which nodes are “Modeling Elements” (ME) connected by the dependency relationship (Wires). Modeling element have a name (string, unique in a given execution), a property Map (Attribute, String), and a set of links (linkType, linkId, lingTarget).

Each time something changes in the ASM, Apam will store in Mongo DB the ME that changed. Apam manages three “collections” (RDBMS tables in Mongo vocabulary): Modeling Element creation, Wiring operations and Property changes.

# Modeling Element Collection

Each time a modeling element is created or deleted, a document is added in the “ME” collection :

For creation : <name, time, op, properties>  
For deletion : : <name, time, op>

* Name is the name of the element.
* Time is the time passed in miliseconds counted since the epoch
* Op is operation, and can be either “created” or “removed”
* Properties is a list <attr value> with attr being the name of a property, and value its value as a String. Note that integers, enumerations and their respective set type are represented as String in the LDAP format.

The implementation for creating the document is as follows:

BasicDBObject created = new BasicDBObject("name", comp.getName())

.append("time",System.currentTimeMillis())  
.append("op","created");

for (Map.Entry<String, Object> e : comp.getAllProperties().entrySet()) {

created.append(e.getKey(), e.getValue().toString());

}

ME.insert(created);

Note that each attribute is individually stored, and therefore can be directly fetched.

Example: creation of the entity "CapteurTemp".

{ "\_id" : ObjectId("5107e03eaacd3fce82732139"), "name" : "CapteurTemp", "time" : NumberLong("1359470654891"), "op" : "created", "description" : "Un capteur de température ", "shared" : "true", "spec-name" : "CapteurTemp", "instantiable" : "true", "singleton" : "false" }

# Property Collection

When a property is a set on a given ME, a document is inserted in the “Attr” collection:

For attribute creation (first value setting): <name, time, “op”:”added”, attribute, value, properties>  
For attribute deletion: <name, time, “op”:”removed”, attribute, oldValue, properties>  
For attribute change: <name, time, “op”:”changed”, attribute, value, oldValue, properties>

* Name is the name of the element
* Time is the time in miliseconds since the epoch
* Op represents the operation an can either be “created”, “removed” or “changed”.
* attribute is the name of the attribute that has changed,
* value the new value of the attribute (a string)
* OldValue is the previous value of the attribute (empty otherwise)
* Properties, has the same characteristics as in ME.

DBCollection ChangedAttr = db.getCollection("Attr");

BasicDBObject newVal = **new** BasicDBObject("name", comp.getName())

.append("time", System.*currentTimeMillis*())

.append("op", "changed").append("attribute", attr)

.append("value", newValue).append("oldValue", oldValue);

**for** (Map.Entry<String, Object> e : comp.getAllProperties().entrySet()) {

newVal.append(e.getKey(), e.getValue().toString());

}

ChangedAttr.insert(newVal);

Example : the attribute “OS” of entity “TestAttr-0” has been changed from “Linux” to the set of values "Android, Linux, IOS".

{ "\_id" : ObjectId("5107e03faacd3fce82732173"), "name" : "TestAttr-0", "time" : NumberLong("1359470655109"), "op" : "changed", "attribute" : "OS", "value" : "Android, Linux, IOS", "oldValue" : "Linux", "autoSet" : "Z-3, Z-2", "s1i" : "5", "testEnumere" : "v2", "S1toS2Final-Bool" : "true”, "spec-name" : "STestAttr", "instantiable" : "true", "singleton" : "false", "location" : "living", "s1b" : "true", "inst-name" : "TestAttr-0" }

# Link Table

When a relationship is created or deleted between two MEs, Apam creates the following document:

<name, time, relType, relId, relTarget>

* Name the origin of the link.
* relType, the type of the relationship (most often “wire”).
* relId the identifier of that link; unique within the same ME.
* “removed” : relTarget or “added” : relTarget.

DBCollection ChangedLink = db.getCollection("Links");

BasicDBObject newLink = **new** BasicDBObject("name", wire.getSource().getName())

.append("time", System.*currentTimeMillis*())

.append("linkType", "Wire").append("linkId", wire.getDepName())

.append("added", wire.getDestination().getName());

ChangedLink.insert(newLink);

Example : the entity "MainApam-0" is wired to the component "APAM-Instance" through the dependency called "apam".

{ "\_id" : ObjectId("5107e03eaacd3fce82732147"), "name" : "MainApam-0", "time" : NumberLong("1359470654969"), "linkType" : "Wire", "linkId" : "apam", "added" : APAM-Instance" }

# Limitations :

Changes other than those mentioned above are not recorded (ASM changes), more specifically:

* Changes in the base level : Method calls and message exchanges,
* Changes in the meta level (changes in the definition of properties, dependencies, constraints)

As a workaround, it is possible to declare attribute , i.e. properties for which the value is the value of a field in the Java program. It allows to publish as an attribute value (e.g. temperature), the value returned by a sensor for example. Each time the value is changed, the attribute is changed, and it is stored in the DB.

Dynamic changes at meta level are possible but not allowed currently.

# Implementation

These records are stored in a Mongo database, in the collections “Attr”, “ME” and “Links”.

HistMan is a manager, and therefore works as all managers. To record an history, you must start a Mongo database server (install and start it on a machine “M” on port “P”), and you must have HistMan in your distribution.   
The configuration is found, as for other managers, in the “conf” directory of the Felix executable, in a file named “root.HISTMAN.cfg”.

This file can contain the following pairs <attribute, value>:

* DBName the name of the Mongo database that will contain your history, default value is assumed ApamRootHistory.
* DBUrl the url of the Mongo server, optionaly followed by the port number, localhost is assumed in case not specified.
* dropCollectionsOnStart= true | false. If true, the collections are dropped each time HistMan will be launched.

This file is optional. If not found, the following values are assumed:

DBName=ApamRootHistory  
DBUrl=localhost  
dropCollectionsOnStart= true

At the time HistMan is started, if the database is not found, histMan unregisters and stops. In that case, there is no history, thus no impact in the apam performance.

Being a manager, a different “conf.HISTMAN.cfg” file can be associated to each composite “conf”. It means that different histories can be recorded in different databases for each different composite, application or silo.