# The Drools Reasoner QuickGuide

02-04-12

## Components

The Drools Reasoner has only 1 component, but, for developing and testing purposes, it can be useful to use other projects which are provided in these preliminary versions of the component.

The Drools Reasoner is being developed under this svn location: <http://forge.universaal.org/svn/new_services/sandbox/mllorente/>

In this folder are placed the following projects:

* droolsDeps: third-party dependencies needed to run Drools Reasoner in an OSGi runtime. These bundles can be added in the pax-runner configuration dialog using the button “Add Bundle…”.
* drools.caller.test: Testing project intended to validate the service profiles provided by the main bundle.
* drools.publisher.test: Dummy publisher created for feeding the context bus with random information. It publishes an event from a device Sensor (with the ontology “Sensor” from universAAL.ontologies “ont.PhWorld”. The event always follows the same RDF triple: “Sensor#concreteSensor -> PROP\_HAS\_MEASUREMENT -> int\_value. Where *int\_value* is a random integer in the range between 0 and 40 in a uniform distribution.
* drools.reasoner: Main functional components. It creates rule engine as a singleton(i.e.: it is not possible having more than one instances of the rule engine running with each class loader) with lazy construction (i.e.: until the RulesEngine class is not called the first time, it is not created in the runtime). This project provides the API to manage with the rules engine and has all the needed components for providing the service and being aware of the context events.
* drools.suscriber.test: Component with test the Consequences publication in the context bus.
* droolstest: Drools tester. It has not component of the universAAL middleware so, in principle, can be used in a generic Apache Felix runtime without any universAAL component. Used for demonstrations and rules testing.
* ont.drools: The project where are all the ontologies related with the drools reasoner placed. Necessary for a correct execution of all the components (except droolstest, of course).

## Usage

For simplification, only the usage of the drools.reasoner, the drools.publisher.test and partially ont.drools will be explained. It is assumed that the reader of this document has enough knowledge about subscribing to context events, managing with ontologies and calling services.

* drools.publisher.test: when this bundle is started, two events are automatically sent to the context bus. One with int\_value = 18 ; and other with int\_value = 19. After that, the bundle asks if the user want to start the auto-publishing mode which publishes random measurements. For accepting that, “Y” or “y” must be introduced by keyboard and after press Enter.
* ont.drools: the most important ontology in the current release is the Consequence ontology. A consequence is a set of information about a specific rule firing. In other words, a consequence is the smaller unit of information which encapsulates the information of a fired rule. A consequence has an unique identifier and a set of properties. The properties of a consequence are key-value pairs that represent the data of the fired rule and the meaning of that.
* drools.reasoner: as it has been introduced above, the drools reasoned has all the components needed for providing the reasoning features that it offers. As the drools.reasoner is a singleton, it can be accessed statically from the class, calling

RulesEngine.getInstance(bundleContext)

Since the RulesEngine is not created automatically when the bundle is started, it is necessary to provide the bundleContext one time to allow the correct managing of resources and utilities provided by the middleware.

From the moment that the instance of the rules engine is available, context events can be sent to it, by using the method insertContextEvent(ContextEvent e).

When a rule is fired, it must use the RulesEngine class to publish a consequence, that is performed by calling the method RulesEngine.getInstance().publishConsequence(String uri, String[] keys, String[] values), where uri is the uri of the new consequence, keys is an array representing all the keys-side of the properties and values is an array representing all the values-side of the properties of the consequence. In the future, the way of passing the key-values pairs to the helper method will be changed for making it more consistent. The idea is to send them using a structure like a table.

In this pre-release, the rules are statically loaded to the knowledge base when the Rules Engine is created. The rules are place in a file called “reasoner.drl” which is under the folder “src/main/resources”. The only way to modify rules is changing this file an re-building the project with maven, which is a relatively easy task for universAAL developers.

All the remaining features of the drools.reasoner are not implemented completely in this pre-release or are not relevant for a universAAL application developer.

## Issues and contact

There are many known issues in this early version. Because of that, doing a list with all of them would be a boring task that is nothing but a waste of time.

Rather than that, if you find an issue or have troubles for get working any of the components you can contact me via mail ([mllorente@tsbtecnologias.es](mailto:mllorente@tsbtecnologias.es)) or skype (mallorentec). It will be a pleasure to know a user of this magnificent component ☺ .