

Jimys

J2EE Is Monitoring Your System

User Manual

Document Versions

Version	Reference	Evolution Cause	Date
1.0.0	UM1.0.0	Creation	11/08/2005
1.0.1	UM1.0.1	Complete	12/08/2005
1.0.2	UM1.0.2	Complete	17/08/2005
1.0.3	UM1.0.3	Correction	22/08/2005
1.0.4	UM1.0.4	Correction	23/08/2005
1.0.5	UM1.0.5	Correction	25/08/2005
1.0.6	UM1.0.6	Modification	29/08/2005
1.0.7	UM1.0.7	Complete	31/08/2005
1.0.8	UM1.0.8	Complete	01/09/2005
1.0.9	UM1.0.9	Complete	03/09/2005
1.0.10	UM1.0.10	Complete	03/09/2005
1.0.11	UM1.0.11	Correction	05/09/2005
1.0.12	UM1.0.12	Correction	06/09/2005
1.0.13	jimys_usermanual1.0.13	Update for delivery	20/10/2005

Table of Contents

1	INTRODUCTION	1
2	ABOUT JIMYS	2
2.1	WHAT'S JIMYS ?	2
2.2	WHY JIMYS ?	2
2.3	WHAT ARE THE ADVANTAGES TO USE JIMYS ?	2
2.4	WHO MAY USE JIMYS ?	2
2.5	INSIDE JIMYS	3
3	JIMYS SOURCES	4
3.1	INTRODUCTION	4
3.2	FOLDER STRUCTURE	4
3.3	JIMYSJ2EE FOLDER	6
3.3.1	Modules	6
3.3.2	Compile JimysJ2EE	6
3.3.3	Scripts	7
3.4	JIMYSPROBES FOLDER	7
3.4.1	Folders	7
3.4.2	Important files	7
3.4.3	Compiling the sources	9
4	GETTING STARTED	10
4.1	ENVIRONMENT	10
4.2	JONAS INSTALLATION	10
4.2.1	Java Messaging Service configuration	10
4.3	DATABASE INSTALLATION	10
4.4	JIMYSJ2EE INSTALLATION	10
4.5	INSTALLING THE JIMYS PROBES MODULE	10
5	CONFIGURATION GUIDE	12
5.1	CONFIGURE JIMYSJ2EE	12
5.1.1	The Analysis-Decision Engine	12
5.1.2	The Mediator	13
5.1.3	The DBManager	15
5.2	CONFIGURE THE JIMYS PROBE MODULE	17
5.2.1	Pump configuration:	18
5.2.2	Admin configuration:	18
5.2.3	List of the deployed probes:	18
5.2.4	Probe specific configuration:	19
5.2.5	The different probes	20
5.2.6	Configuration file of the GenericJmx probe	22
5.3	22
5.4	CONFIGURE THE HIBERNATE	23
5.5	RULES	24
5.5.1	Application data	24
5.5.2	Defined rules	24
5.5.3	Write your own rules	25
5.5.4	Validate the rules	26
6	START/STOP JIMYS	27
6.1	USING THE SCRIPTS	27
6.2	MANUALLY	27
6.2.1	Deploy the application	27
6.2.2	Start the pump and probes	27
6.2.3	Undeploy the application	27
7	USING THE WEB USER INTERFACE	28
7.1	INTRODUCTION	28

7.2	GETTING CONNECTED	28
7.3	CONFIGURATION MENU	28
7.3.1	Mediator	28
7.3.1.1	Introduction.....	28
7.3.1.2	Main	29
7.3.1.3	Detail configuration.....	30
7.3.1.4	Mediator configuration	31
7.3.2	Rule Engine	32
7.3.2.1	Fire Rules	32
7.3.2.2	Activate Deactivate Rule	32
7.3.2.3	Show configuration.....	33
7.4	MONITORING MENU.....	34
7.4.1	Show all data	34
7.4.2	Cluster status	35
7.4.3	Monitor a Probe in a chart.....	38
7.4.3.1	Main	38
7.4.3.2	Charts.....	40
7.4.4	Monitor a Rule in a chart.....	41
7.4.4.1	Main	41
7.4.4.2	Charts.....	43
7.4.5	Live Cluster check.....	44
7.4.6	Cleanup database.....	45
7.5	ALARMS	46
7.6	CARTOGRAPHY.....	47

1 Introduction

This document describes the User Manual of the Jimys tool. The goal is to lead the user through the installation process to ensure the system is installed correctly. It explains also how the different functionalities are used.

2 About Jimys

2.1 What's Jimys ?

Jimys is a monitoring tool for the J2EE applications that aims to facilitate the operator's life. It offers :

- an advanced error detection feature through a alarms generation mechanism and a global view of the managed system status.
- a tracking activity of the managed nodes through some graphical charts.

2.2 Why Jimys ?

Such tools are more and more required because the systems become very complex. Humans meet huge difficulties to take care of all the components at the same time. And when you deploy your application on a J2EE cluster, the requirement is far more critical. There are a very large number of events to examine that are generated at different levels (system, JVM, J2EE, application) and on several nodes for a cluster configuration.

2.3 What are the advantages to use Jimys ?

Jimys tackles these issues, the relevant informations for the supervision are gathered in a single graphical interface for the whole managed system, the data are updated on the fly. When a fault occurs, the operator is notified. The time of intervention is reduced and since the MTTR (Mean Time To Repair) is reduced, the MTBF is improved. Moreover, Jimys is based on a rules engine that leverages its flexibility. Such tool allows to automatically sort, and analyse the huge amount of events that may be raised by the system, in order to provide the administrator with only the relevant information. It also allows customizing the error detection rules in order to adapt them at your administration policies. Even the actions can be enhanced to implement, in addition of the alarm generation for example, a mail sending or a reconfiguration task on the managed system.

2.4 Who may use Jimys ?

All the users who have to deal with complex monitoring tasks, for example :

- to supervise a clustered J2EE application
- to supervise a set of standalone J2EE applications

2.5 Inside Jimys

Jimys provides both the probes infrastructure enabling the retrieving and the gathering of the events on the managed nodes and a centralized analysis/decision module being able to process these events on the fly and yields alarms or status for the operator. A lot of probes are already available to supervise different levels on the managed nodes :

- system level with a CPU probe, a memory probe, a network probe and a disk probe
- JVM level with a JDK 1.5 probe - J2EE level with a JMX probe polling the JOnAS's MBeans or a JOnAS log's probe retrieving the log content
- application level with either a generic JMX probe or a generic log probe

Moreover, a heartbeat mechanism checks both whether the managed system is alive and whether the probes are alive.

The processing module of Jimys relies on an original architecture based on Drools (engine rules). Through the inference motor, Jimys allows to define high level rules for the filtering/aggregating/correlation of the events and even to implement some automatic behaviours that interact with the managed system (supervision loop).

The centralized part of Jimys is a J2EE application on top of JOnAS. The rules engine (Drools) is accessed through a RAR connector and the alarms are stored in a database such as MySQL.

3 Jimys sources

3.1 Introduction

Jimys is delivered as a package with the different modules and configuration files. The configuration has to be customized.

3.2 Folder structure

There are 2 main folders which contain the sources and binaries of the 2 main applications:

- The “**Analysis-decision system module**”, the “**User interface module**” and “**Storage module**” which are located in the folder “*jimysJ2EE*”. This should be deployed on a standalone machine which monitors the other machines. This is a heavy application which needs good hardware.
- The “**Probe module**” is located in the folder “*jimysProbes*”. This application should be deployed on the monitored machines. This is a lightweight application and can run in parallel of some other java processes.

The distribution contains the following files and folders:

```
+ jimysJ2EE
  - build.xml
  +bin
    - installRun.sh
    - install.sh
    - javaCheck.sh
    - run.sh
    - stop.sh
    - uninstall.sh
  +ADEngineEJB
    + etc
    + conf
    + lib
    + src
  +MediatorEJB
    + etc
    + conf
    + lib
    + src
  +JimysCommons
    + src
  +DBCommunication
    + etc
    + conf
    + lib
    + src
  +VisualizeJimys
```


- + WEB-INF
- + resources
- + lib
- + src
- +DroolsRar
 - + etc
 - + lib
 - + src
- + *jimysProbes*
 - startProbes.sh
 - build.xml
 - + dist
 - BootJimys.jar
 - JimysProbes.jar
 - +conf
 - config.xml
 - logging.properties
 - mlet.xml
 - mletHardware.xml
 - mletJonas.xml
 - mletJonasHardware.xml
 - monitor.properties
 - monitorHardware.properties
 - monitorJonas.properties
 - monitorJonasHardware.properties
 - + probes
 - joramProbeConfig.xml
 - probeconfig-414.xml
- + lib
 - 3rd party libraries
- + libMini
 - mx4j.jar
 - mx4j-jmx.jar
 - mx4j-remote.jar
 - mx4j-tools.jar
 - xercesImpl.jar
 - xml-apis.jar

3.3 JimysJ2EE folder

3.3.1 Modules

The following are the different modules being part of the distribution. The different folders contain subfolders with their libraries, deployment descriptor, sources and configuration files. The structure of the different folders can change in order to the needs of the modules,

- ADEngineEJB
 - This module contains everything which is necessary to work with the rule engine
- MediatorEJB
 - This module corresponds to the mediator
- DBCommunication
 - This module does the database access and contains the business logic. (EJB and hibernate)
- VisualizeJimys
 - This corresponds to the graphical user interface (struts application)
- DroolsRAR
 - This is the resource adapter with the Drools rule engine
- JimysCommons
 - This contains all the files that are shared between the modules
- bin
 - Contains all installation and run scripts

3.3.2 Compile JimysJ2EE

To compile JimysJ2EE use the build.xml in the JimysJ2EE's root folder. These are the main targets:

- make_dist
 - Using this target, the complete distribution is build and ready to be deployed with the scripts. This target builds the different modules and copies them with their libraries and configuration files into the distribution folder "dist".
- clean_all
 - This cleans all class folders from the different modules and erases the output directory.
- update_dist
 - Use this target if you have changed a configuration file and don't want to rebuild the complete project. The configuration file is copied in the distribution folder and is ready to use.
- build_all
 - This target builds the different modules in the output folder.

The build.xml contains other targets which are not explained in detail. These targets are for the different modules, and can be used for example to compile, precompile the jsp, ...

3.3.3 Scripts

- installRun.sh
 - The installRun script installs the different packages, libraries and configuration files in JOnAS. After the installation, JOnAS starts and Jimys is deployed.
- install.sh
 - Installs the different packages, libraries and configuration files in JOnAS
- javaCheck.sh
- run.sh
 - Starts JOnAS and deploys Jimys.
- stop.sh
 - Stops JOnAS
- uninstall.sh
 - Remove the different packages, libraries and configuration files from JOnAS.

3.4 JimysProbes folder

3.4.1 Folders

- dist
 - contains the libraries of the jimys probes. This folder normally contains only 2 jars.
- conf
 - contains the configuration files of the probes and the JMX server
- lib
 - contains all the necessary libraries in order to compile the distribution jars from the sources.
- libMini
 - contains the minimum of libraries in order to be able to run the probes. These libraries are the mx4j libs and also the xml implementation libraries. Attention: You must also have the lib “BootJimys.jar” in the **dist** directory to start the probes!

3.4.2 Important files

- startProbes.sh
 - startup script to start the probe module. You can modify this script in order to take advantage of a different configuration (monitorXY.properties).
- build.xml
 - The ant script to build the binaries from the sources. The target “bootJar” produces the binary “BootJimys.jar”. The target “jar” produces the binary distribution “JimysProbes.jar” of the probes and the pump. This jar is designed to be on a central place (<http>

server) and is downloaded dynamically by the probe module when this one starts. This has the advantage that you only need to replace the jar on a single place. (The url is in the monitorXY.properties which points to the mletXY.xml). The real url is coded in the mletXY.xml file which is also placed on the central place. Please consult chapter 4.5 for installation instructions or the JMX specification.

- Config.xml
 - Configuration file of the mx4j server. You can modify the port which is opened for the jmx server.
- Logging.properties
 - Config file of the logger.
- mletXY.xml
 - These files define the mlet configuration. This file is intended to be stored somewhere on a web server. But it is not mandatory. This file defines all the MBeans which are loaded by the JMX¹ server. For example the “mletJonas.xml” defines only the probes (MBeans) which are able to monitor jonas parameters like log, jonas and joram. You can also use the “mletJonasHardware.xml” and decide only to start the jonas probes. But the specific libraries for the hardware probes will also be downloaded by the probe module. In this file, you define where the libraries are located (codebase). Normally, this is a url which points to a web server url but you can also let the url point to the file system. That means you don't need to download the libraries dynamically from a webserver but you can also copy them somewhere on the file system. This reduces the network load on startup.
- monitorXY.properties
 - This file defines the mlet url and also which probes are started. Furthermore, it defines some parameters of each probe. Attention: this file is specified in the startup script “startProbes.sh”. This is one of the most important files for the probe module. For the different syntax, please look in chapter 4.5 for a detailed explanation of this file.
- joramProbeConfig.xml
 - Configuration of the joram probe. The syntax of this file is similar to the Jonas probe².
- probeconfig-414.xml
 - Configuration of the jonasjmx probe. The syntax of this file is similar to the Jonas probe³.

Remark: The “XY” in the name of the monitoringXY.properties and mletXY.xml files is an alias for the different example files in the “conf” directory. When “XY” is replaced by “Jonas” you use the jonas monitoring files. Using the Hardware files let you monitor the hardware architecture. Using the files where “XY” is replaced by nothing (mlet.xml), you have all the available probes. But be prepared to see some errors when using these files because some monitored resources are not running.

¹ See the Java specification of JMX

² This probe has been developed by BULL SA.

³ This probe has been developed by BULL SA.

3.4.3 Compiling the sources

There is an ant build script in the root of the directory.

The different targets are:

- clean
 - removes the compiled files and also the distribution jar files
- compile
 - compiles the sources
- jar
 - produces the “JimysProbes.jar” in the dist folder.
- bootJar
 - produces the “BootJimys.jar” in the dist folder.
- startProbes
 - starts an MBeanServer and deploys all the probes defined in the mletXY.xml specified in the monitoringXY.properties file. Consult the “build.properties” file to change the path to the “monitoringXY.properties” file.
- Client
 - starts a client thread which connects to a pump to retrieve the data. This application illustrates how to connect to the pumps and retrieve the data. You can extend this class in order to use the information collected by the pump. Consult the “build.properties” file to change the host and port to connect to.

4 Getting started

4.1 Environment

The following tools and software are necessary to run Jimys correctly

- Host running JimysJ2EE
 - Linux Kernel 2.4+
 - Java Virtual Machine 1.4.8+
 - Apache Ant 1.6.2+
 - JOnAS 4.6+
 - MySQL 3.2354+ or PostgreSQL 7.4+ or another JDBC compliant product
- Clusternodes
 - Linux Kernel 2.4

4.2 JOnAS Installation

Please refer to the JOnAS installation manual at <http://jonas.objectweb.org>.

4.2.1 Java Messaging Service configuration

Jimys needs 2 topics and 1 queue to work correctly. The name of them must be defined in the property files “mediator.properties” and “engine.properties”. See the JOnAS documentation to configure the JMS topics and queue in JOnAS.

4.3 Database Installation

Jimys can be use in combination with for example a MySQL or PostgreSQL database.

- MySQL :
 - Please refer to the MySQL installation manual at <http://www.mysql.org>.
- PostgreSQL:
 - Please refer to the PostgreSQL installation manual at <http://www.postgresql.org>.

4.4 JimysJ2EE Installation

1. Extract jimysJ2EE on your disk.
2. Make “ant make_dist” with the build.xml to create the distribution
3. Next step is the configuration.

4.5 Installing the Jimys probes module

1. Define a central machine where you want to put all the libraries and mlet config file. This machine needs to have a running http server. You can use a JOnAS server which does the monitoring for this purpose.
2. Put the mletXY.xml file on the HTTP server in a way that it is accessible through : <http://host:9000/mletXY.xml>
3. Put all the libraries which are referenced by the mlet file to the same place or to the url which is specified in the mlet file.
4. Copy the "jimysProbes" folder on all the machines which should be monitored. You can delete the "lib" folder if you want to save disk space. The only jars you absolutely need are "dist/bootJimys.jar" and the libs in the "libMini" folder. You can also delete the source and build directory.
5. Now the configuration of the probe module. If you used in step 2 for example "mletJonas.xml", please use now also the monitorJonas.properties. But it is not mandatory.
6. Adapt the different urls in the monitorXY.properties in order to match your network settings (jonas port, protocol, ...) and file system settings (the absolute path to the probe, jmx configuration files)
7. Adapt the "startProbes.sh" script in order to take the right monitorXY.properties file as argument.
8. Launch the "startupProbes.sh" script and you hopefully see no errors.

Remarks:

- The default port for the JMX server is 1999
- If you try to monitor a JOnAS server or another JMX server, this server needs to run before the probes are started.
- Please be sure to have defined the JAVA_HOME environment variable

5 Configuration guide

5.1 Configure JimysJ2EE

5.1.1 The Analysis-Decision Engine

The “engine.properties”-file contains different properties which must be configured. Different properties must be the same as specified during the configuration of JOnAS to work correctly. Here are the configuration properties.

1. Main configuration

- clustername
 - Description: the name of the cluster to monitor
 - Value: jimys
- engine.conf.fire.rule.after
 - Description: defines after how many received events the rules are fired
 - Value: 10
- engine.conf.fire.interval
 - Description: defines in milliseconds after how many milliseconds the rules are fired. If the value is 0 then the timer is inactive
 - Value: 0
- engine.rar.connection
 - Description: this parameter indicates the JNDI name of the resource adapter for the rule engine.
 - Value: JimysDrools
- engine.conf.use.listener
 - Description: Setting this property to true, the system informs the user when an object has been asserted/retracted from the rule engine and counts the events into the working memory.
 - Value: false

2. Database topic connection

- database.java.naming.factory.initial
 - Description: Java naming factory
 - Value: com.sun.jndi.rmi.registry.RegistryContextFactory
- database.java.naming.provider.url
 - Description: naming provider url
 - Value : rmi://localhost:1099
- database.remote.home
 - Description: Remote home of the DBManager
 - Value: DBManagerHome

- database.topic.name
 - Description: the jms topic the DBManager subscribes
 - Value: dbtopic
- database.topic.connection.factory
 - Description: the name of the topic connection factory
 - Value: JTCF
- database.connection.username
 - Description: topic username
 - Value: anonymous
- database.connection.password
 - Description: topic password
 - Value: anonymous

3. Rules

The following properties specify the rules which can be activated or deactivated at runtime. Each rule which is listed here gets an activator and needs an entry in the file with the rules.

- engine.rule.rulesetName_ruleName
 - Description: indicates that the rule “ruleName” in the ruleset “rulesetName” is activable / deactivable at runtime
 - Value: true or false

5.1.2 The Mediator

The configuration is done in the “mediator.properties”-file.

1. Main configuration

- mediator.connect.pump
 - Description: the JMX object name of the pump which runs on the nodes
 - Value:
imys:mbean=org.objectweb.jimys.monitoredSystem.pump
- Mediator.pump.connect.local.home
 - Description: the local stateful session bean home which is used by the mediator to connect to the different pumps. For each pump, a new session is created.
 - Value: PumpConnectHome_L

2. Parser topic

- mediator.java.naming.factory.initial
 - Description: Java naming factory
 - Value: com.sun.jndi.rmi.registry.RegistryContextFactory
- mediator.java.naming.provider.url
 - Description: naming provider url
 - Value : rmi://localhost:1099
- mediator.mdb.topic.name
 - Description: the name of the topic the parser subscribes to.

- Value: topic
- mediator.mdb.topic.connection.factory
 - Description: the connection factory of the topic
 - Value: JTCF
- mediator.mdb.topic.connection.username
 - Description: the username to access the topic
 - Value: anonymous
- mediator.mdb.topic.connection.password
 - Description: the password to access the topic
 - Value: anonymous

3. Engine queue

- engine.java.naming.factory.initial
 - Description: Java naming factory
 - Value: com.sun.jndi.rmi.registry.RegistryContextFactory
- engine.java.naming.provider.url
 - Description: naming provider url
 - Value : rmi://localhost:1099
- engine.mdb.queue.name
 - Description: the name of the queue the engine binds to
 - Value : queue
- engine.mdb.queue.connection.factory
 - Description: the connection factory for the queue
 - Value: JQCF

4. Available pumps

Here it is here possible to specify a list of know hosts, where a pump is running. With the help of this list, it is not necessary to specify each host to connect, but Jimys will show a selectable list to the user with the hosts.

- mediator.pumpX
 - Description: this indicates a host with a pump. X is a number ascending from 0 to the number of hosts
 - Example :


```
mediator.pump0 hostAB:1999
mediator.pump1 hostCD:1999
...
```
 - Value: hostname:port

5. Cluster check

These properties are used to make a live cluster check. This check will return the health of the whole cluster

- mediator.attribute.livecheck.name
 - Description: the attribute name of the pump MBean to receive the heartbeat
 - Value : HeartBeat

6. Registered probes

Attribute of the pump MBean to receive a list of the registered probes.

- mediator.attribute.registered.probes.name
 - Description: the JMX attribute name to get the list of all the registered probes.
 - Value : RegisteredProbes

7. Change probe sample time

- mediator.attribute.get.probe.sample.time.name
 - Description: the name of the JMX attribute to get a probe sample time
 - Value : getProbeSampleTime
- mediator.attribute.set.probe.sample.time.name
 - Description: the name of the JMX attribute to set a probe sample time
 - Value : setProbeSampleTime

5.1.3 The DBManager

The “dbmanager.properties”-file contains the configuration of the DBManager.

1. database connection

- database.local.home
 - Description: name of the local stateful session bean home
 - Value: DBManagerHome_L
- database.remote.home
 - Description: name of the remote stateful local session bean home
 - Value: DBManagerHome
- database.java.naming.factory.initial
 - Description: Java naming factory
 - Value: com.sun.jndi.rmi.registry.RegistryContextFactory
- database.java.naming.provider.url
 - Description: naming provider url
 - Value: rmi://localhost:1099

2. engine connection

- engine.web.session.name
 - Description: The name of the home of the engine stateless session bean
 - Value: EngineWebEJBHome
- engine.time.session.name
 - Description: The name of the stateless session bean home used to initialize the timer to fire the rules
 - Value: RuleFireTimerEJBHome
- engine.java.naming.factory.initial
 - Description: Java naming factory
 - Value: com.sun.jndi.rmi.registry.RegistryContextFactory
- engine.java.naming.provider.url
 - Description: naming provider url
 - Value: rmi://localhost:1099

3. mediator connection

- mediator.web.session.name
 - Description: name of the mediator stateful session bean home
 - Value: MediatorWebEJBHome
- mediator.java.naming.factory.initial
 - Description: Java naming factory
 - Value: com.sun.jndi.rmi.registry.RegistryContextFactory
- mediator.java.naming.provider.url
 - Description: naming provider url
 - Value: rmi://localhost:1099

5.2 Configure the Jimys probe module

We will only explain the mletXY.xml file and monitorXY.properties file. The config.xml file can be modified according the mx4j documentation. And the “probeconfig-414.xml”, configuration file of the jonas probe⁴.

mletXY.xml :

In this file you define all the MBeans which are deployed on the client machines (monitored machines). This file is intended to be accessible through a http url. But you can also copy it to the file system.

One entry defines one MBean.

```
<mlet      code="mbeanImplementationClassName"
           name="mbeanName"
           archive="libs,you,need,to,download"
           codebase="urlWhereTheLibsAreLocated">
```

- code
 - The class you name here is instantiated by the MBean server.
- name
 - The Object name of the MBean. With this name you are able to access the MBean.
- archive
 - A list of jar files which are mandatory in order to instantiate the MBean. That means, if you instantiate a class in the MBean implementation specified by “**code**”, this class must be in one of the archives defined here.
- codebase
 - The url where the MBean server is able to download the jar files specified in the **archive** attribute.

Remarks:

- Do not bother that the file is not a valid xml file. (No root node is specified).
- You can define also mbeans here which are not used afterwards by the probe module. It is possible to have a large mlet.xml where all the possible mbeans are defined but in the monitor.properties you specify which one are used by the pump. Attention, when you specify mbeans here, the mbean server will instantiate them. So it is possible that you receive some errors on start up. For example when you specify the CJDBC mbean, this probe will immediately try to establish a connection to the cjdbc server. But when no server is running you will see an error. This error will not cause any trouble for the other mbeans.

monitorXY.properties :

⁴ This probe was developed by BULL SA. Please ask BULL SA. For additional information

We will explain how you configure the JimysProbes and which possibilities you have.

Global configuration of the MBean server:

- mlet.file
 - defines the url where the mlet file is found. This has to be a valid URL. It can be on the file system or on a web server.
- mlet.name
 - defines the object name of the mlet bean in the MBean server when it is deployed. This property may stay like this, no need to change.
- mx4j.config
 - the url where the configuration file of the mx4j server is stored.

Remark: From now on we will not repeat the prefix “org.objectweb.jimys.monitoredSystem” for each property.

5.2.1 Pump configuration:

- pump.name
 - the object name of the pump mbean in the mbeanserver
- pump.compress
 - enable/disable the compression on the pump. Enabling the compression reduces the amount of data send over the network but increases the cpu load on both sides. When you use a client or the mediator of JimysJ2EE you must also enable the decompression to be able to see the data.
- pump.sampletime
 - Specify the sample rate of the pump. This is the time between two messages sent from the pump to the client (mediator). This parameter is arbitrary, if not specified the sample rate is 5 seconds.

5.2.2 Admin configuration:

- admin.name
 - the object name of the admin mbean in the mbeanserver.

5.2.3 List of the deployed probes:

- probes
 - the list of the probes which will be registered in the pump. Do not confuse with the mbeans which will be deployed on the mbeanserver. These are specified in the mlet file. Each name of the list is separated by a “;”. The names specified may be extended and customized. But when you specify a new name in

this list, you also have to specify the probe specific properties which will be:
`"org.objectweb.jimys.monitoredSystem.probe"+"NEW_NAME"!`

5.2.4 Probe specific configuration:

Each probe has specific properties which must be defined. Some are mandatory, some are arbitrary.

Mandatory:

- name
 - the object name of the probe in the mbeanserver. This is the same name specified for the mbean in the mlet file !
- attributes
 - the list of attributes which are collected by the pump. This is a list of attribute names which is available through the mbean interface of the mbean probe. The different attributes are separated by a “;”.
- sampleTime
 - this is the time the pump waits to collect the attributes for the next time.

Arbitrary:

- init
 - some probes must be initialized to work correctly. For example the jonasjmx probe must be initialized to receive the correct url of the mbeanserver of jonas. After initialization, the probe is able to connect to the other mbean server and collect the data. You have to look at the mbean interface of the probe to see how many attributes you must specify in the init property. The init method of the probe mbean interface has only “java.lang.String” parameters. The number of parameters depends on the probe. Each parameter is separated by a “;”

5.2.5 The different probes

In general: All the possible attributes of the probes which may be monitored must be in the interface of the probe mbean. If you want to add a new attribute, you have to add the method of in the interface.

- Cpu
 - This probe can monitor the load of the CPU. We chose not to use the CPU probe of the lewys probes because we had some problems to calculate the load. This probe only works on linux OS. The load is calculated by taking two snapshots of the data in the proc file system (/proc/stat) for a given time between the snapshots. The load is calculated : $((data2 - data1)/(time2 - time1)) * 100$
- Network
 - This probe analyses the network traffic (Rx, Tx). The load is calculated the same way than the cpu. The probe is based on the lewys probes.
- Memory
 - This probe calculates the memory usage of the system. The probe is based on the lewys probes.
- Filesystem
 - This probe calculates the IO load of the file system. The probe is based on the lewys probes.
- Cjdbc
 - This probe collects information of the cjdbc. Attention, this probe must be initialized. The probe is based on the lewys probes.
- Apache
 - This probe collects the log of the apache web server. This probe needs to be initialised in order to be able to find the log file. Attention, this probe opens the file and reads the whole file. If you have a large log file you may encounter performance problems.
- Jonas
 - This probe is a wrapper around the j2ee probe developed by BULL SA⁵. This probe needs to be initialised. Attention, this probe in jimys is *deprecated* because you should use the jonasjmx probe which is based on the genericjmx probe.
- Log
 - This probe analyses the monolog output and can be used in parallel of the jonasjmx probe. This probe must be initialised. The probe registers a jmx listener in monolog which runs in a second mbeanserver.
- Cartography
 - This probe analyses the hardware of the system. The data sent by this probe is constant (as long as you do not have any hot pluggable architecture). Therefore the sample rate should be very high. This probe is based on the lewys probes and works only on linux systems. (Preferable kernel 2.4.x)

⁵ Please ask BULL SA for additional information.

- Sunjvm
 - This probe analyses the sunjvm. This probe is *deprecated* because you should use the genericjmx probe. This probe permits to monitor the specific mbeans inside a jvm 1.5.
- Genericjmx
 - This probe is a generic probe which allows probing attributes of mbeans deployed on other mbeanservers. In order to have multiple instances of the genericjmx probe, you must have different configuration files (see next chapter). Furthermore, you must specify a different “name” in the mlet file but always the same “code”. That means, you can have 2 entries in the mlet file which have as code “.GenericJmx” but 2 different object names. This probe must be initialised. The first parameter of the init method specifies the mbeanserver url of the mbeanserver to connect to. The second parameter is the location of the configuration file and the third parameter is the name of the probe which must be different from the other probes. A good example of using the genericjmx probe is the joram probe.
- Jonasjmx
 - This probe collects the data from a jonas instance. This probe must be initialized. If you want to monitor multiple jonas instances on the same machine you must specify in the mlet file 2 elements of this probe with different names. You must also change the first and third parameters of the init method. The configuration file may stay the same if you want to retrieve the same data.
- GenericLogFile
 - This probe collects the different events (alert, warning, info, ...) produced by any application which writes a log file. The log file must have different levels of messages and associate a timestamp to each event in the log file. For exotic log files, it is probably necessary to customize the “GenericLogFile”.
- Joram
 - Please consult the example of the joram probe which uses the GenericJmx probe.

5.2.6 Configuration file of the GenericJmx probe

The syntax of the configuration file is identical to the configuration of the Jonas probe⁶.

Here is a small example:

```
<?xml version="1.0" encoding="UTF-8"?>
<probe-config>
  <probe-tasks dateFormat="yyyy_MM_dd_HH_mm" defaultRefreshPeriod="300000"
defaultSamplingPeriod="10000"/>
  <polls-descriptor logdateformat="dd/MM/yyyy HH:mm:ss"
logtimeformat="HH:mm:ss" >
    <poll name="joramDestination">
      <discriminator>Joram:type=Destination,*</discriminator>
      <attribute-list>
        <attribute>
          <attribute-name>NbMsgsDeliverSinceCreation</attribute-name>
        </attribute>
        <attribute>
          <attribute-name>NbMsgsReceiveSinceCreation</attribute-name>
        </attribute>
        <attribute>
          <attribute-name>NbMsgsSendToDMQSinceCreation</attribute-name>
        </attribute>
      </attribute-list>
    </poll>
  </polls-descriptor>
</probe-config>
```

Remarks:

- The attributes of the elements “probe-tasks” and “polls-descriptor” are not used by JimysProbes. They are only there to be compatible with the J2EE probe⁷.
- The element “poll” defines the name of the mbean to monitor. You can use regular expressions like “*:j2eeType=JDBCDataSource,*”. This means that all the mbeans matching this expression will be pulled to collect the information.
- The “attribute” elements define which attributes of the mbean are pulled.
- You can have multiple “poll” elements which define different mbeans in the same mbeanserver to pull. Like the sample configuration file of the jonasjmx probe.

5.3

⁶ This probe was developed by BULL SA. Please ask BULL SA for further information.

⁷ This probe was developed by BULL SA

5.4 Configure the Hibernate

The configuration of hibernate in Jimys is done using 2 files:

- hibernate.cfg.xml : Hibernate configuration
- mapping.hbm.xml : Database mapping file

hibernate.cfg.xml

Different properties must be specified in this configuration file. All the information is related to the configuration of hibernate and the database. No mapping information should be specified in this file.

- session-factory name
 - The JNDI name of the session factory which is used to bind it in the JNDI tree.
- connection.datasource
 - The JNDI datasource name. This name is the datasource name of the database configured in JOnAS. Default : jdbc_1
- Dialect
 - The SQL dialect to use. Please see the hibernate documentation for a complete list of valid values. (For example : "org.hibernate.dialect.PostgreSQLDialect")
- show_sql
 - Show the executed SQL strings at runtime.
- mapping resource
 - The mapping file name to use.

mapping.hbm.xml

The mapping of all the java beans is done in this file. You should not modify this file! But it depends on how the database analyzes the SQL string. The tables are created by the cmp2 beans the first time. Now it is possible that the database interprets the SQL string case sensitive. So it is possible that you have to see the table names after creation with a database tool and change the table names in the mapping.hbm.xml file.

5.5 Rules

The definition of the rules is done in the “rules.drl”-file.

5.5.1 Application data

Application Data is data in the rule engine that is used by all the rules. This data is not checked against a rule when it is inserted into the working memory. The following data is necessary for the engine:

- pumps
 - to check the pump health at runtime
- status
 - to check the clusterstatus at runtime

The following data is optional:

- type of RuleActivator
 - This data is used to enable activation or deactivation of a rule. The value must be an instance of RuleActivator. The value should have the same name as the rule name. This is the convention used. If a ruleactivator is defined, then it is necessary to define the same rule in the mediator.properties file. When data is not defined here, then it will not be possible to use the activator.

5.5.2 Defined rules

The system has some defined rules. These rules can be used or modified. Each of these rules can be activated /deactivated at runtime. To handle this, they all need a ruleactivator object.

Following rules are defined

- pumpChecker
 - This rule checks every 40 seconds if the registered pumps are alive.
- checkProbesPumps
 - The rule checks the health of the whole cluster. The rule is fired after having received an heartbeat from a pump.
- r1_cpu_alarm_80
 - This rule generates an alarm if it receives a Cpu event and the CPULoad attribute is higher than 80%
- r1_cpu1
 - This rule writes an entry in the database if it receives a Cpu event and checks the attribute CPULoad
- r1_mem1
 - This rule writes an entry in the database if it receives a Memory event
- r1_net1
 - This rule writes an entry in the database if it receives a Net event and checks the attribute RxNetworkLoad

- r1_fs1
 - This rule writes an entry in the database if it receives a FileSystem event
- r1_jl1
 - This rule writes an entry in the database if it receives a JonasLog event
- r1_cjdbc1
 - This rule writes an entry in the database if it receives a Cjdbc event
- r1_jonas1
 - This rule writes an entry in the database if it receives a JonasMBean event
- r1_joram1
 - This rule writes an entry in the database if it receives a Joram event
- r1_cart1
 - This rule writes the actual cartography in the database
- r1_cpu_trans
 - This rule write an alert in the database if it receives from the same host a Cpu event and a JonasMBean event where the CPULoad is higher than or equal to 10% and the JonasMBean hitsCount of the cache is higher than or equal to 2000
- killer
 - This rule retracts every object that is longer than 60 seconds in the working memory

5.5.3 Write your own rules

To write your own rules it is necessary to understand the methodology adapted by the rule engine vendor to write the rules and how the rule engine works. Please consult first the documentation about Drools at www.drools.org

To write the different rules, different helper can be used:

- MDBConnector
 - The connector helps to write the result of a rule in the database
- PumpCheckHelper
 - This helper checks if the pumps are still sending their data
- ClusterStatusHelper
 - This helper defines a method to check the whole cluster status and write it in the database if a change has occurred at runtime
- CartographyHelper
 - The CartographyHelper is used to extract the cartography from a cartography event and to write the data in the database
- LevelHelper
 - The LevelHelper has a set of define levels which are used to assign an alarm level to an entry in the database.

Following events can be used. To get more details please consult the External Specification in the chapter about interesting parameters to observe.

- AbstractProbe
- Cpu
- Memory
- Network
- JonasMBean
- Joram
- Cjdbc
- Filesystem
- JonasLog
- Admin

The following is what a rule should look like:

```
<rule name="rulesetname_ruleName">
```

Each rule has a rule name and at least one parameter. The convention for the rule name that is used is "rulesetname_rulename".

The condition is optional. More conditions are possible. The expression in the condition clause must be a boolean expression.

The consequence is required. The consequence defines what to do when the conditions are fulfilled. As shown in the example, this rule will write a message to the console and add an entry in the database by using the MDBConnector.

```
</java:condition>
```

5.5.4 Validate the rules

```
<java:consequence>
```

The system contains a rulebase validator to check if the rule file with the rules does not have any semantic error. It ensures not that the rules are doing the right thing.

```
system.addProbe(cpu);
mdb.sendResult("Your Cpu load is over 90 %",
    LevelHelper.Alert,"cpu1","r1");
</java:consequence>
```

```
</rule>
```

6 Start/Stop Jimys

6.1 Using the scripts

The easiest way to start/stop Jimys is to use the scripts that are in the jimysJ2EE folder.

6.2 Manually

6.2.1 Deploy the application

To deploy Jimys from scratch the following steps are necessary

1. Copy contents of jimysJ2EE/dist/apps to \$JONAS_BASE/apps
2. Copy contents of jimysJ2EE/dist/conf/ to \$JONAS_BASE/conf
3. Copy contents of jimysJ2EE/dist/lib to \$JONAS_ROOT/lib/ext
4. Copy contents of jimysJ2EE/dist/rars to \$JONAS_BASE/rars
5. Start JOnAS with : jonas start
6. Deploy Mediator : jonas admin -a Mediator.ear
7. Deploy the Rule Engine : jonas admin -a DroolsRAR.rar
8. Deploy ADEngine : jonas admin -a ADEngine.ear
9. Deploy DBManager : jonas admin -a DBManager.ear

6.2.2 Start the pump and probes

Use the scripts to start the pump and probes

6.2.3 Undeploy the application

To undeploy Jimys

- With JOnAS continuing to run
 1. Undeploy Mediator : jonas admin -r Mediator.ear
 2. Undeploy the Rule Engine : jonas admin -r DroolsRAR.rar
 3. Undeploy ADEngine : jonas admin -r ADEngine.ear
 4. Undeploy DBManager : jonas admin -r DBManager.ear
- With JOnAS stopping
 1. jonas stop

7 Using the Web User Interface

7.1 Introduction

The Web User Interface is based on the JOnAS web interface "jonasAdmin"

7.2 Getting connected

To connect to the Web User Interface, use a web browser that supports Frames and CSS.

URL: <http://host.to.connect.to:port/VisualizeJimys>

To login use the username and password specified in the jonas.properties.

Username: jonas

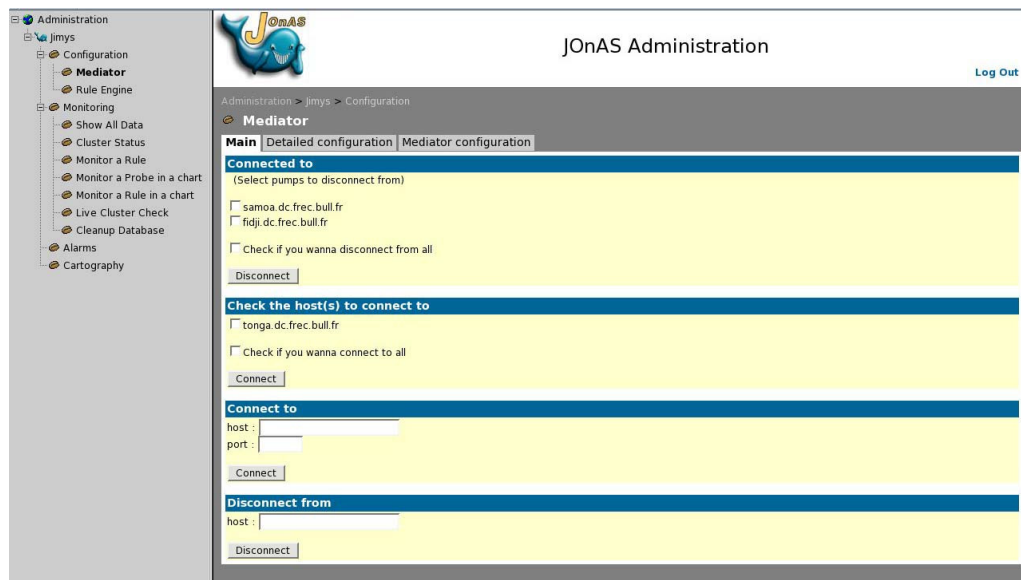
Password: jonas

7.3 Configuration menu

7.3.1 Mediator

7.3.1.1 Introduction

This configuration menu is used to connect/disconnect to the different pumps of the cluster.



It is also possible to change the sample time of the different probes and to show the actual configuration of the mediator

7.3.1.2 Main

In this page, it is possible to see which pumps are connected, to connect to new ones or to disconnect from pumps.

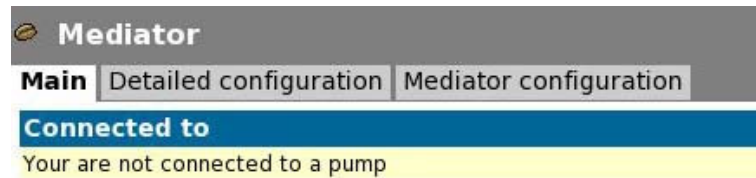
- Connected to

Here are the pumps listed where the mediator is connected to.



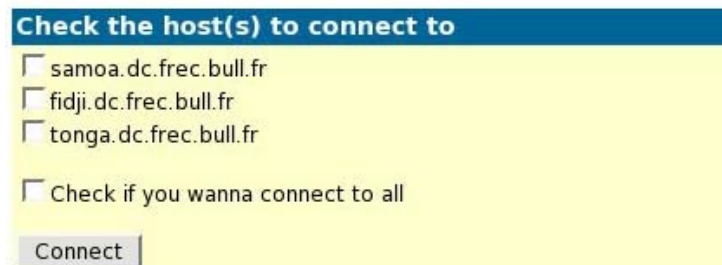
The screenshot shows a window titled "Mediator" with three tabs: "Main", "Detailed configuration", and "Mediator configuration". The "Main" tab is selected. Below the tabs is a section titled "Connected to" with a subtitle "(Select pumps to disconnect from)". This section contains three checkboxes: "samoa.dc.frec.bull.fr", "fidji.dc.frec.bull.fr", and "Check if you wanna disconnect from all". At the bottom of this section is a "Disconnect" button.

Or the following if no connection can be found



The screenshot shows the same "Mediator" window with the "Main" tab selected. The "Connected to" section now displays the message "Your are not connected to a pump".

- Check the host(s) to connect to



The screenshot shows a window titled "Check the host(s) to connect to". It contains three checkboxes: "samoa.dc.frec.bull.fr", "fidji.dc.frec.bull.fr", and "tonga.dc.frec.bull.fr". Below these is a checkbox labeled "Check if you wanna connect to all". At the bottom is a "Connect" button.

Here it is possible to connect to the different pumps which have been specified in the mediator.properties file.

- Connect to



A dialog box titled "Connect to" with a blue header. It contains two input fields: "host :" and "port :". Below the fields is a "Connect" button.

If a new host has been added at runtime in the cluster and should monitor it, it is not necessary to stop Jimys. At runtime, it is possible to add new hosts to monitor by specifying the hostname and port where the probes run.

- Disconnect from



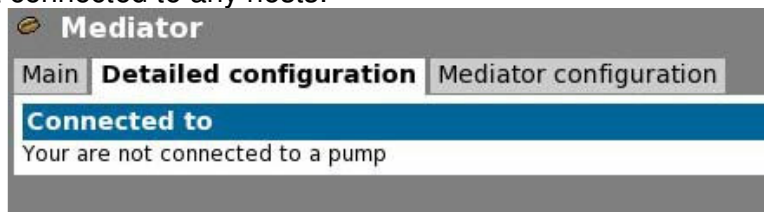
A dialog box titled "Disconnect from" with a blue header. It contains one input field: "host :". Below the field is a "Disconnect" button.

Enter the hostname to disconnect from.

7.3.1.3 Detail configuration

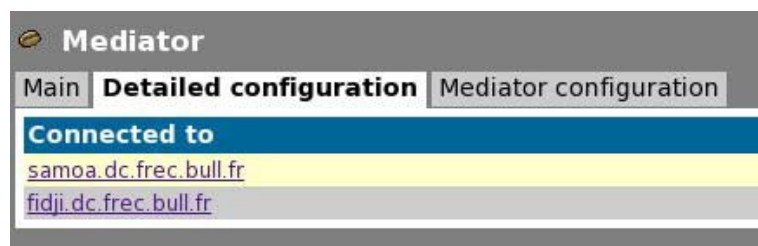
To see the sample time of the different pumps it is necessary to being connected to at least one host.

When not connected to any hosts:



The "Mediator" configuration screen with tabs "Main", "Detailed configuration", and "Mediator configuration". The "Detailed configuration" tab is active. It shows a blue header "Connected to" and the text "Your are not connected to a pump".

If connected to some hosts:



The "Mediator" configuration screen with tabs "Main", "Detailed configuration", and "Mediator configuration". The "Detailed configuration" tab is active. It shows a blue header "Connected to" and two hostnames: samoa.dc.frec.bull.fr and fidji.dc.frec.bull.fr.

After selecting a host, the page will display the different sample times that can be changed.

Mediator			
Main Detailed configuration Mediator configuration			
Connected to			
samoan.dc.frec.bull.fr			
fidji.dc.frec.bull.fr			
Probe Name	Current Sample Time	New Sample Time	
ujf:mbean=network	3000		Change
ujf:mbean=jonasjmx	5000		Change
ujf:mbean=log	3000		Change
ujf:mbean=joramjmx	5500		Change
ujf:mbean=memory	3500		Change
ujf:mbean=cpu	3800		Change
ujf:mbean=fileSystem	7000		Change

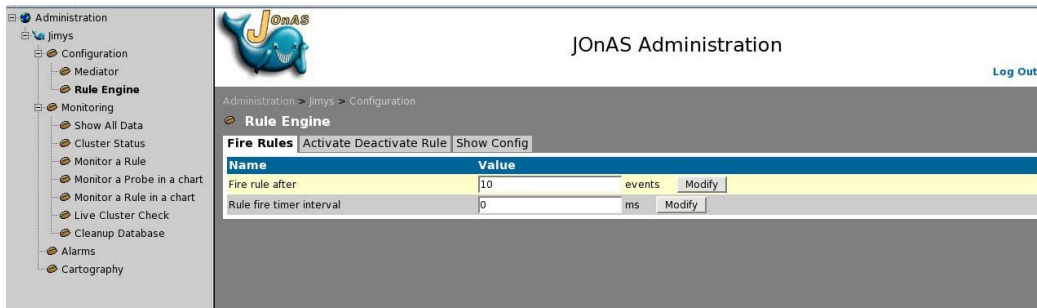
It is only possible to change one parameter at once.

7.3.1.4 Mediator configuration

Mediator	
Main Detailed configuration Mediator configuration	
Name	Value
mediator.pump1	tonga.dc.frec.bull.fr:1999
mediator.attribute.get.probe.sample.time.name	getProbeSampleTime
mediator.attribute.registered.probes.name	RegisteredProbes
mediator.pump0	samoan.dc.frec.bull.fr:1999
mediator.pump.connector.local.home	PumpConnectorHome_L
mediator.mdb.topic.connection.password	anonymous
mediator.mdb.topic.connection.factory	JTCF
mediator.mdb.topic.connection.username	anonymous
mediator.connect.pump	ujf:mbean=org.objectweb.jimys.pump
engine.java.naming.factory.initial	com.sun.jndi.rmi.registry.RegistryContextFactory
engine.mdb.queue.name	queue
mediator.pumps	3
engine.mdb.queue.connection.factory	JQCF
mediator.attribute.livecheck.name	HeartBeat
mediator.java.naming.factory.initial	com.sun.jndi.rmi.registry.RegistryContextFactory
mediator.java.naming.provider.url	rmi://localhost:3099
engine.java.naming.provider.url	rmi://localhost:3099
mediator.mdb.topic.name	topic
mediator.attribute.set.probe.sample.time.name	setProbeSampleTime
mediator.pump2	fidji.dc.frec.bull.fr:1999

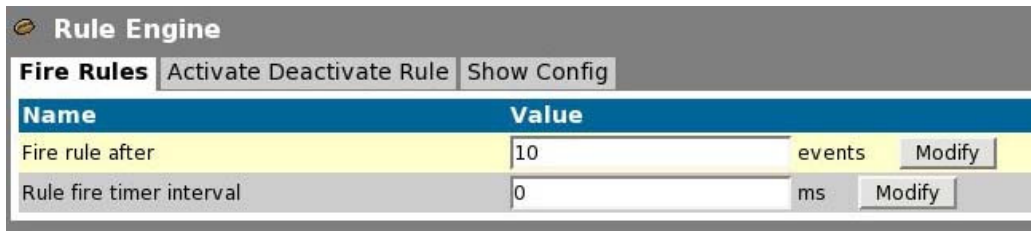
In this page, it is possible to consult the configuration of the mediator which corresponds to the mediator.properties file.

7.3.2 Rule Engine



7.3.2.1 Fire Rules

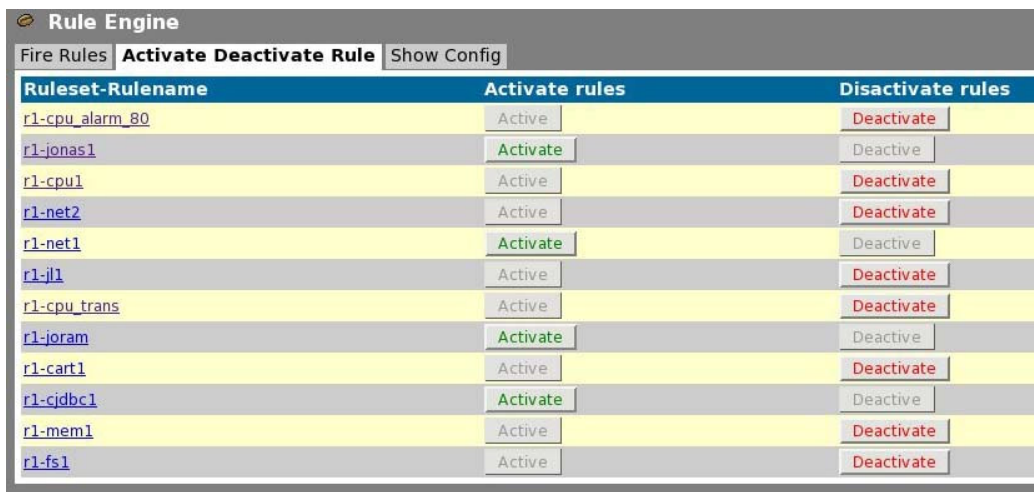
This page shows the configuration parameters of the Analysis-Decision Engine that can be changed at runtime.



If the parameters are changed at runtime, the changes will be lost when restarting the system, if they have not been added in the engine.properties file.

7.3.2.2 Activate Deactivate Rule

This page contains the rules which can be activated and deactivated.



To see the detailed information of a rule, it is only necessary to click on the name of the rule, and the detail is shown to the user.

```

● RuleName :
    r1_cpu_alarm_80

● Condition(s) :
    [Condition: r1_cpu_alarm_80.isActive()]
    [Condition: ((Integer)cpu.getAttributeByKey("CPULoad")).intValue()>=80]

● Consequence :
    [Consequence: MDBConnector mdb = new MDBConnector();
    mdb.addProbeAttribute(cpu,"CPULoad",((Integer)cpu.getAttributeByKey("CPULoad")).toString()); mdb.sendResult("ALERT : Your CPU is
    "+((Integer)cpu.getAttributeByKey("CPULoad")).intValue().LevelHelper.ALERT +"cpu_alarm_80","r1"); mdb = null; ]

```

7.3.2.3 Show configuration

This page shows the configuration of the Analysis-Decision Engine. It is not possible to change the configuration at runtime.


Rule Engine

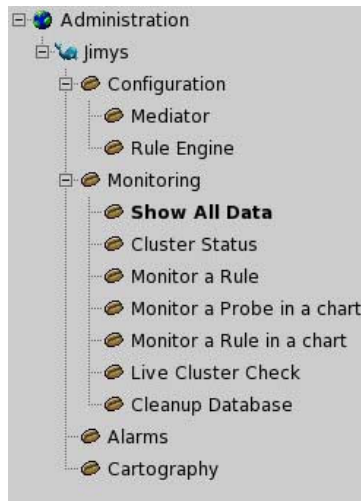
[Fire Rules](#)
[Activate](#)
[Deactivate Rule](#)
[Show Config](#)

Name	Value
database.topic.connection.factory	JTCF
clustername	jimys
database.java.naming.provider.url	rmi://localhost:3099
database.remote.home	DBManagerHome
engine.rar.connection	SAMPLERAR
database.topic.connection.password	anonymous
database.java.naming.factory.initial	com.sun.jndi.rmi.registry.RegistryContextFactory
database.topic.connection.username	anonymous
database.topic.name	dbtopic

This page does not list the parameter that can be found under Fire Rules.

7.4 Monitoring menu

7.4.1 Show all data



Consulting the page “Show All”, the user sees all messages in the database.

Show All Data		
Show All	Detailed Result	
Time Stamp	Result Message	Level
1123834984862	Your RxNetworkLoad is 24	monitoring
1123834985288	Filesystem to db	not_specified
1123834985348	Your Memory is 98	monitoring
1123834985627	Jonas Mbean received	note
1123834985640	Jonas Mbean received	note
1123834985653	Jonas Mbean received	note
1123834985719	Jonas Mbean received	note
1123834985731	Jonas Mbean received	note
1123834985561	Jonas Mbean received	note
1123834985825	Jonas Mbean received	note
1123834985666	Jonas Mbean received	note
1123834985745	Jonas Mbean received	note
1123834985838	Jonas Mbean received	note
1123834985597	Jonas Mbean received	note
1123834986066	Jonas Mbean received	note
1123834986444	Jonas Mbean received	note
1123834986524	Jonas Mbean received	note
1123834986605	Jonas Mbean received	note
1123834986666	Jonas Mbean received	note
1123834986918	Jonas Mbean received	note

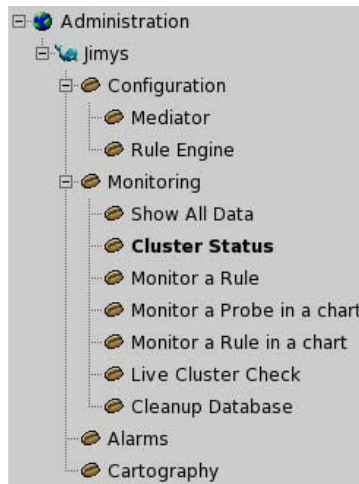
After selecting a timestamp, the user is redirected to the “Detailed Result” and sees the detail of the message.

Show All Data		
Show All	Show Result Detail	
ProbeIdentifier	Attribute Name	Attribute Value
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=network	RxNetworkLoad	24

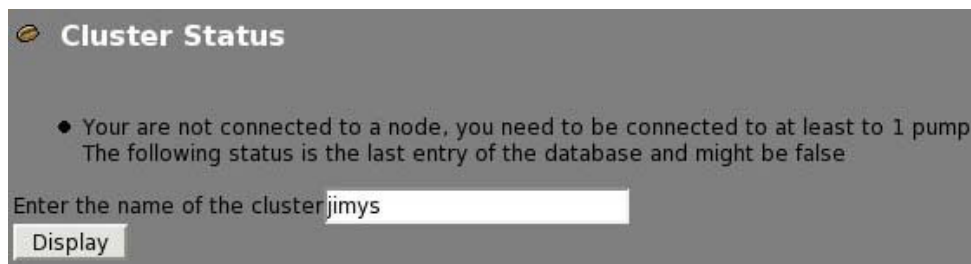
Here are the different probe attributes listed which belong to the message. The probe identifier contains the hostname, domain, and instance to know exactly from which node of the cluster the event came.

7.4.2 Cluster status

Under cluster status, it is possible to check the state of the cluster.

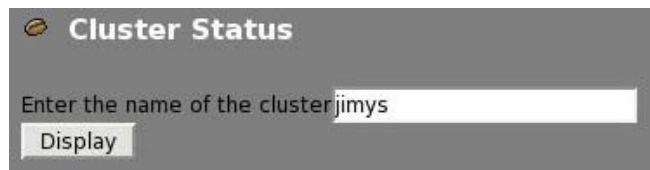


To get real time values, it is necessary to be connected to the different cluster nodes otherwise old data from the database is shown.



The following steps are necessary to get the status:

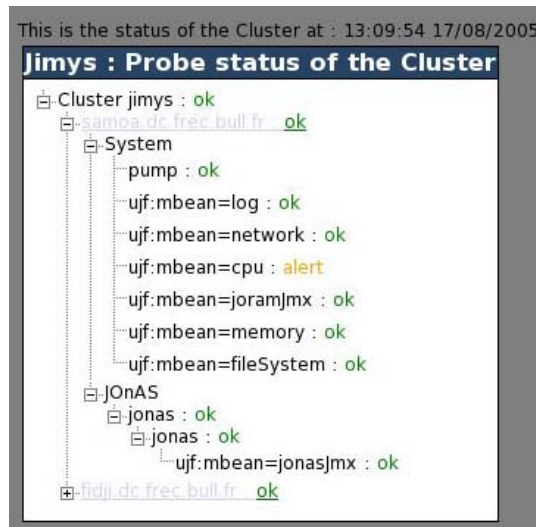
1. Enter the name of the cluster



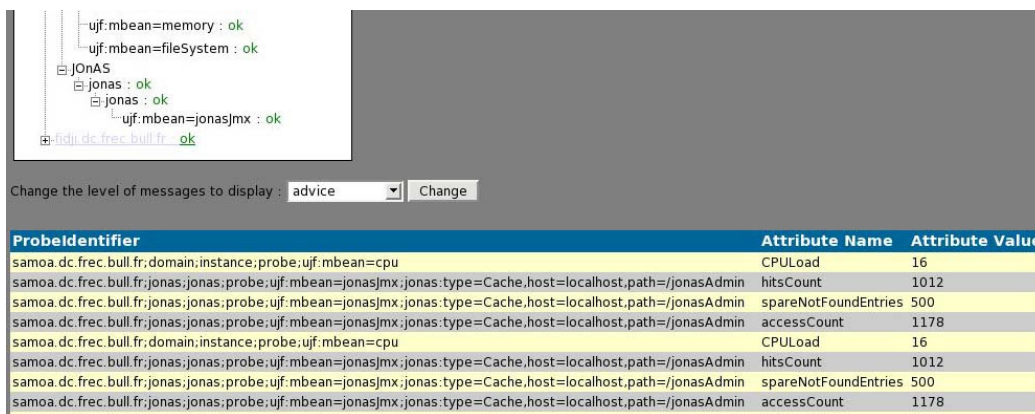
2. Select the branch of the tree to have a more detailed view



3. Select a hostname to see more detailed information about a specific host



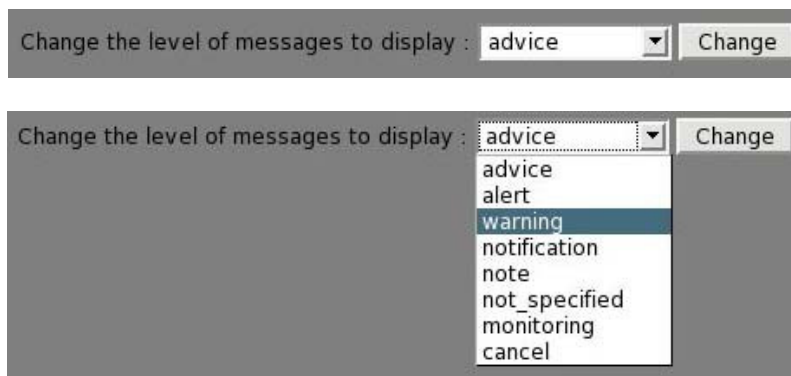
The detailed information



Change the level of messages to display :

ProbeIdentifier	Attribute Name	Attribute Value
samoia.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu	CPUload	16
samoia.dc.frec.bull.fr;jonas;jonas;probe;ujf:mbean=jonasJmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	hitsCount	1012
samoia.dc.frec.bull.fr;jonas;jonas;probe;ujf:mbean=jonasJmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	spareNotFoundEntries	500
samoia.dc.frec.bull.fr;jonas;jonas;probe;ujf:mbean=jonasJmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	accessCount	1178
samoia.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu	CPUload	16
samoia.dc.frec.bull.fr;jonas;jonas;probe;ujf:mbean=jonasJmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	hitsCount	1012
samoia.dc.frec.bull.fr;jonas;jonas;probe;ujf:mbean=jonasJmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	spareNotFoundEntries	500
samoia.dc.frec.bull.fr;jonas;jonas;probe;ujf:mbean=jonasJmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	accessCount	1178

4. To change the level of the messages that should appear, it is necessary to change the level and validate.

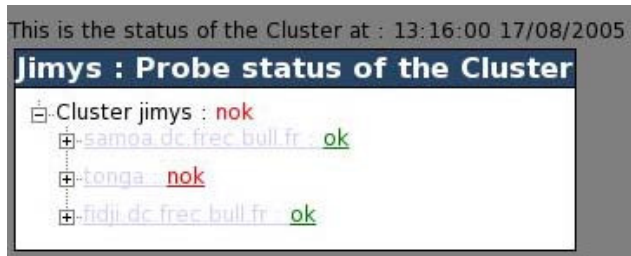


Afterwards it is necessary to select the host in the tree and the messages with the selected level will be displayed

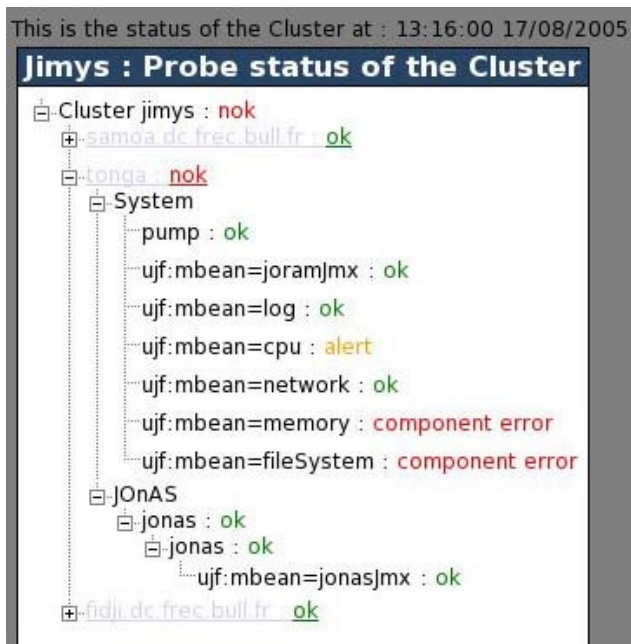
5. In a case of an error on a host, the following state is shown



The administrator sees that the cluster has a problem. Selecting the cluster node, he sees more information about the host that has a problem.

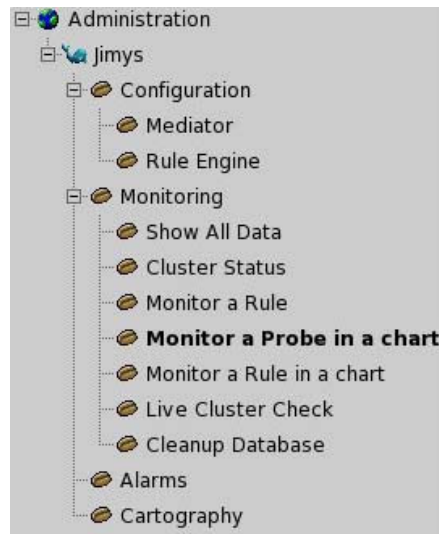


Selecting the host, he sees even more detail about which component produces an error.



Now the administrator can react on the problem.

7.4.3 Monitor a Probe in a chart

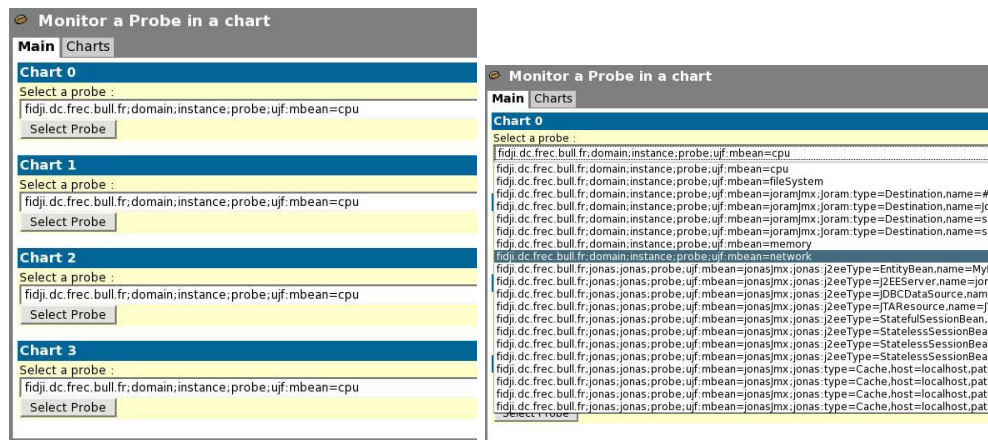


7.4.3.1 Main

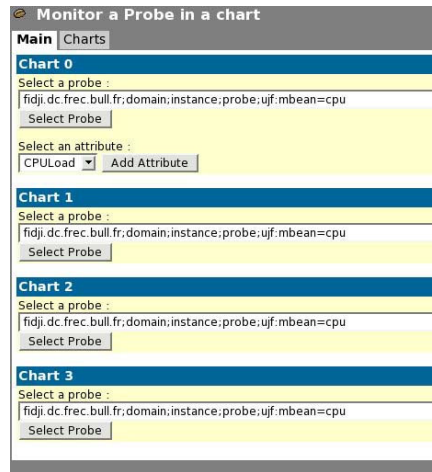
It is possible to have 4 different charts to monitor attributes of the different probes.

The following steps are necessary to monitor a probe attribute

1. Choose a chart and select a probe



2. Then select an attribute of that probe



Monitor a Probe in a chart

Main | **Charts**

Chart 0

Select a probe :
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu
Select Probe

Select an attribute :
CPULoad Add Attribute

Chart 1

Select a probe :
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu
Select Probe

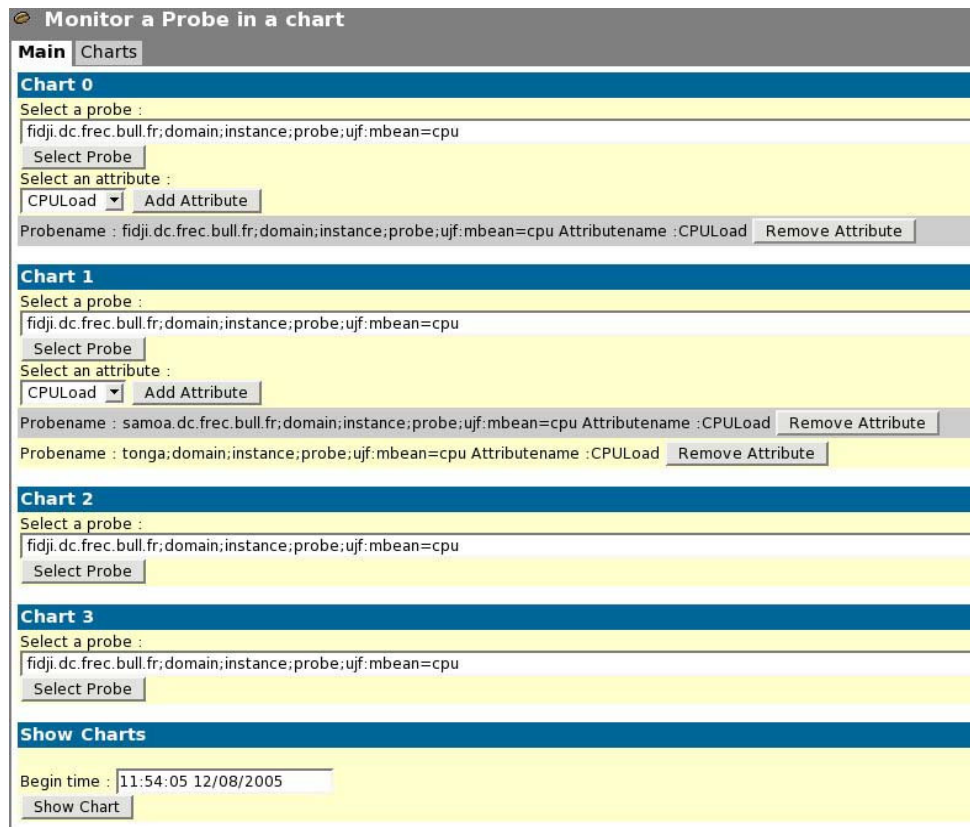
Chart 2

Select a probe :
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu
Select Probe

Chart 3

Select a probe :
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu
Select Probe

3. If necessary, other attributes can be added to the same chart



Monitor a Probe in a chart

Main | **Charts**

Chart 0

Select a probe :
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu
Select Probe

Select an attribute :
CPULoad Add Attribute

Probename : fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu Attributename : CPULoad Remove Attribute

Chart 1

Select a probe :
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu
Select Probe

Select an attribute :
CPULoad Add Attribute

Probename : samoa.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu Attributename : CPULoad Remove Attribute

Probename : tonga;domain;instance;probe;ujf:mbean=cpu Attributename : CPULoad Remove Attribute

Chart 2

Select a probe :
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu
Select Probe

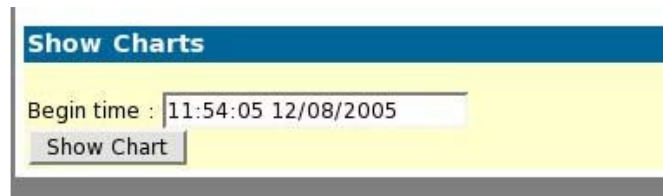
Chart 3

Select a probe :
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cpu
Select Probe

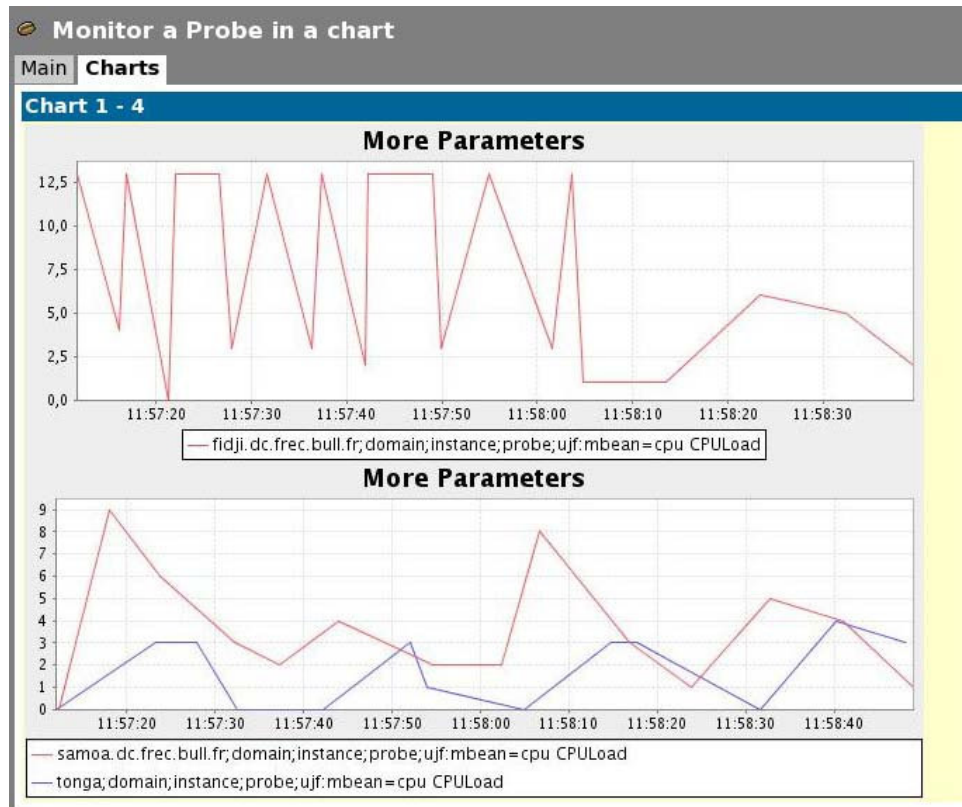
Show Charts

Begin time : 11:54:05 12/08/2005
Show Chart


4. Enter the begin time to start monitoring. If the field is empty, the monitoring starts at the time the “Show Chart”-button has been pressed.



5. The user is redirected to the charts page that will show the different charts.



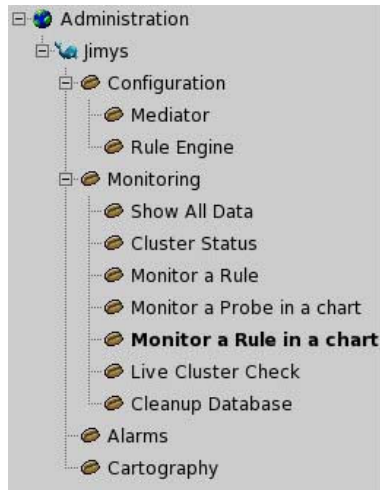
To remove an attribute from a chart, it is only necessary to select the “Remove Attribute”-button of an attribute



7.4.3.2 Charts

If the probes and attributes have been selected then the chart is shown on this page. This page is refreshed every 8 seconds automatically to get the latest states of the attributes.

7.4.4 Monitor a Rule in a chart

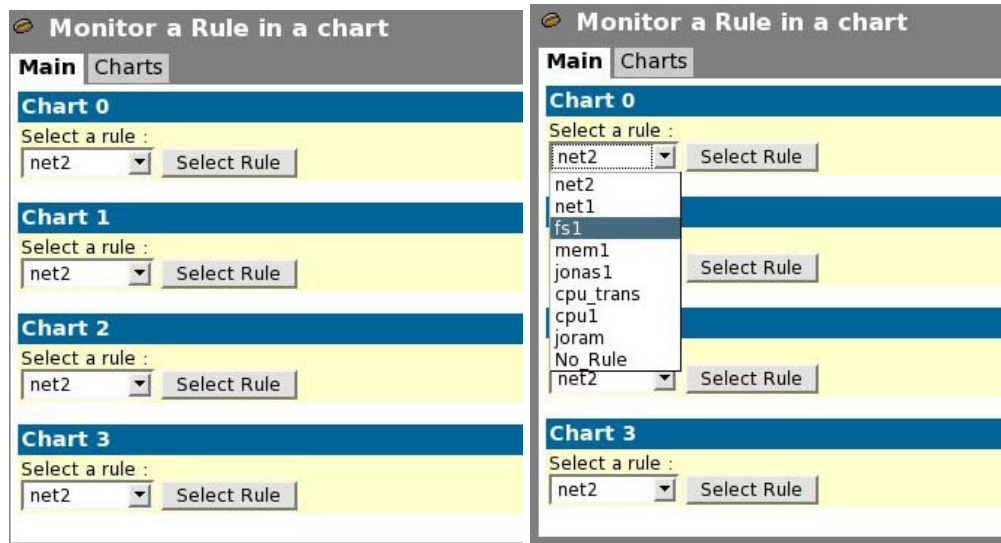


7.4.4.1 Main

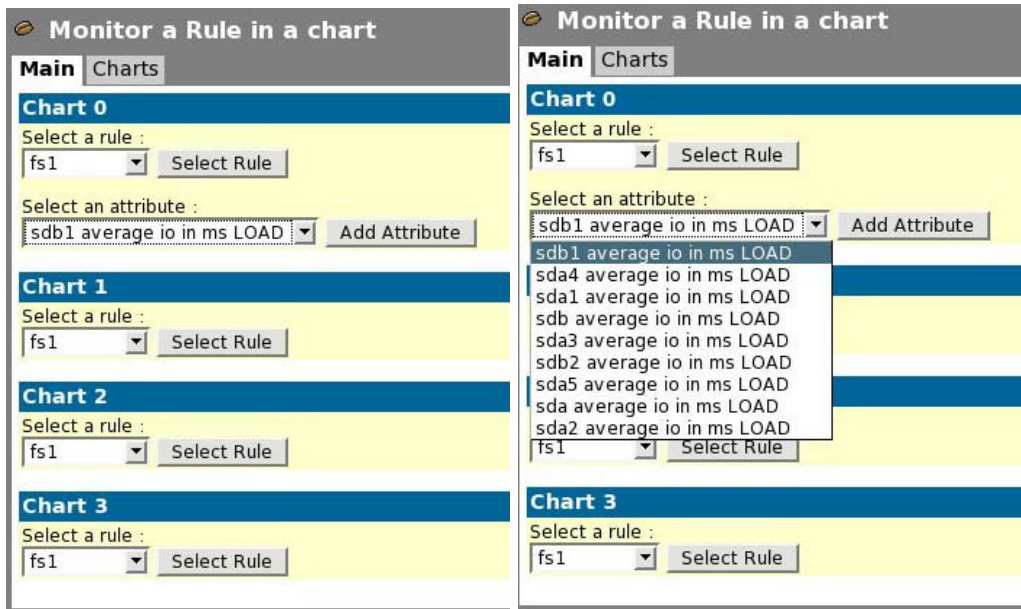
It is possible to have 4 different charts to monitor the result of the different rules.

The following steps are necessary to monitor a rule.

1. Chose a chart and select a rule



2. Select an attribute of that rule



3. Other rules can be added to another chart

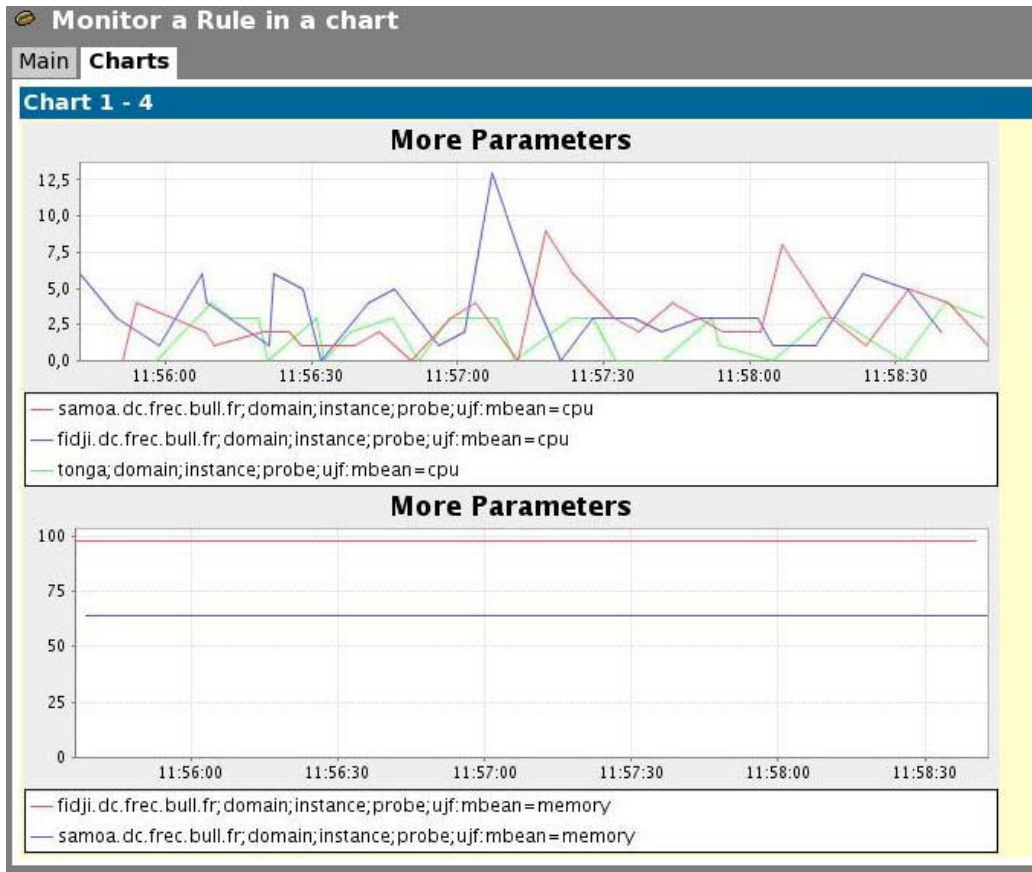


4. Enter the time to start with the monitoring. If the field is empty the monitoring will start at the time the “Show Chart”-button has been pressed.

Show Charts

Begin time :

5. The user is redirected to the chart page that shows the different charts.



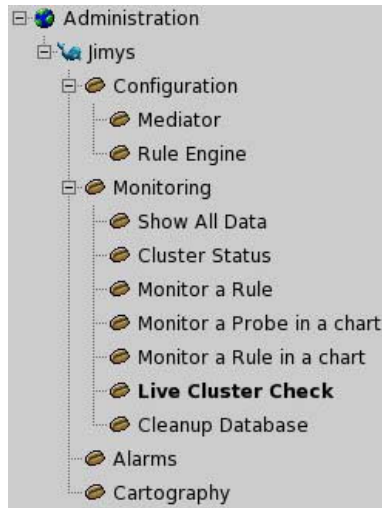
To remove an attribute from a chart it is only necessary to select the “Remove Attribute”-button of an attribute

Rulename : cpu1
 Attributename : CPULoad

7.4.4.2 Charts

If the rules and the attributes have been selected, then the chart is shown on this page. This page is refreshed every 8 seconds automatically to get the latest state of the attributes.

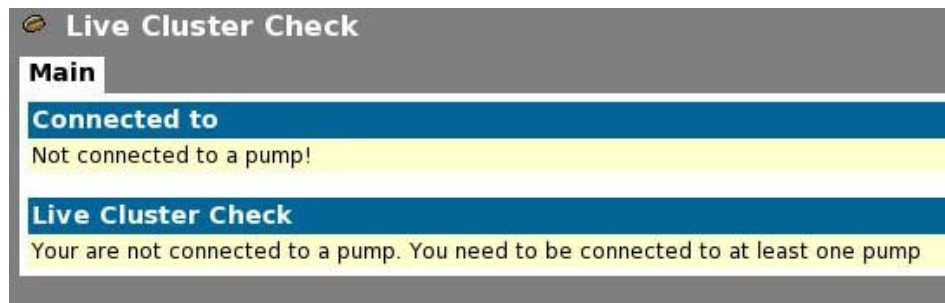
7.4.5 Live Cluster check



The live cluster check helps to get a real time check of the health of the cluster.

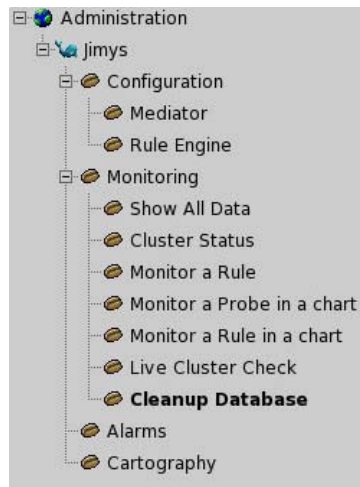
Live Cluster Check		
Main		
Connected to		
<ul style="list-style-type: none"> samoa.dc.frec.bull.fr fidji.dc.frec.bull.fr 		
Live Cluster Check		
samoa.dc.frec.bull.fr		
Time	Probe Identifier	Status
1124277640407	samoa.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=cpu	ok
1124277640407	samoa.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=network	ok
1124277640408	samoa.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=memory	ok
1124277640408	samoa.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=fileSystem	ok
1124277640409	samoa.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=log	ok
1124277640410	samoa.dc.frec.bull.fr;jonas;jonas;probe; ujf:mbean=jonasjmx	ok
1124277640411	samoa.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=joramjmx	ok
fidji.dc.frec.bull.fr		
Time	Probe Identifier	Status
1124277551654	fidji.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=cpu	ok
1124277551654	fidji.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=network	ok
1124277551655	fidji.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=memory	ok
1124277551655	fidji.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=fileSystem	ok
1124277551659	fidji.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=log	ok
1124277551661	fidji.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=cartography	ok
1124277551661	fidji.dc.frec.bull.fr;jonas;jonas;probe; ujf:mbean=jonasjmx	ok
1124277551662	fidji.dc.frec.bull.fr;domain:instance;probe; ujf:mbean=joramjmx	ok

The Mediator needs to be connected to at least one pump to return the health.

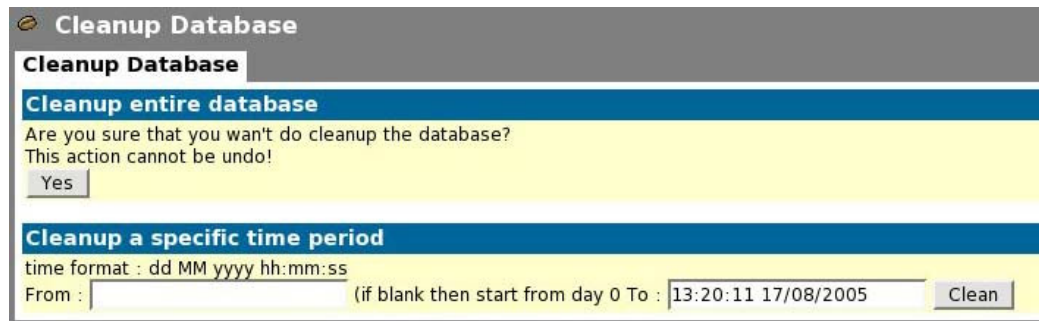


Otherwise it is not possible to get a live cluster status

7.4.6 Cleanup database

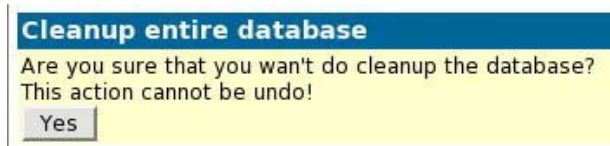


This page helps to clean up the database.



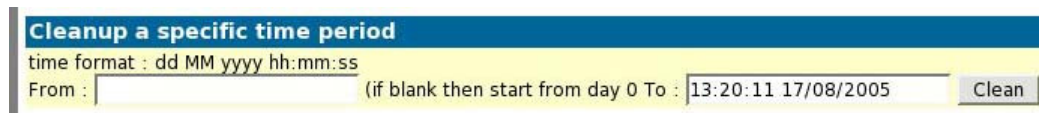
The database can be cleaned up totally or only a specific time period can be erased.

- Total cleanup



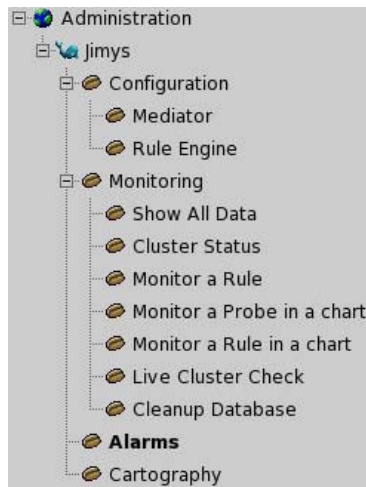
By selecting “Yes” the tables in the database are deleted.

- Time period



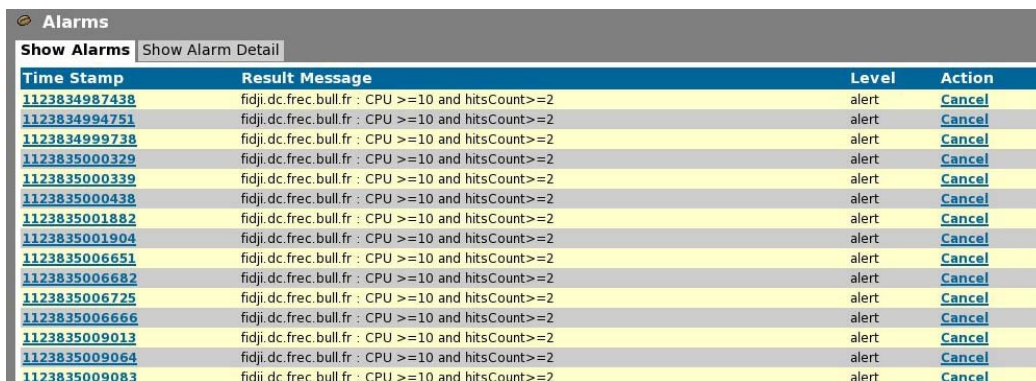
It is necessary to indicate the beginning and ending of the period

7.5 Alarms



- Show Alarms

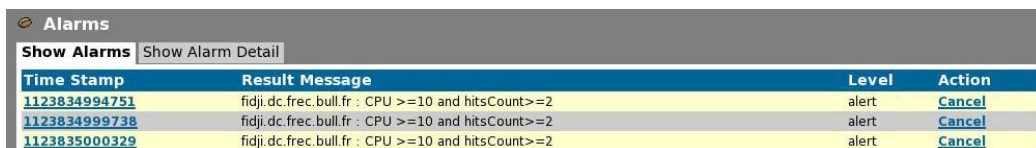
This page shows all the alerts in the database.

A screenshot of the 'Alarms' page. At the top, there are two tabs: 'Show Alarms' (selected) and 'Show Alarm Detail'. Below the tabs is a table with four columns: 'Time Stamp', 'Result Message', 'Level', and 'Action'. The table contains 15 rows of alert data. Each row has a timestamp, a message indicating a CPU usage threshold being reached, the level 'alert', and a 'Cancel' link.

Time Stamp	Result Message	Level	Action
1123834987438	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123834994751	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123834999738	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835000329	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835000339	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835000438	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835001882	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835001904	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835006651	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835006682	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835006725	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835006666	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835009013	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835009064	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835009083	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel

Selecting the timestamp of an alert, the detail is shown on “Show Alarm Detail” page.

An alert can also be cancelled by selecting “Cancel”.

A screenshot of the 'Alarms' page, similar to the previous one, but with only three rows of data remaining. The first two rows have been cancelled. The 'Cancel' links are still present in the 'Action' column.

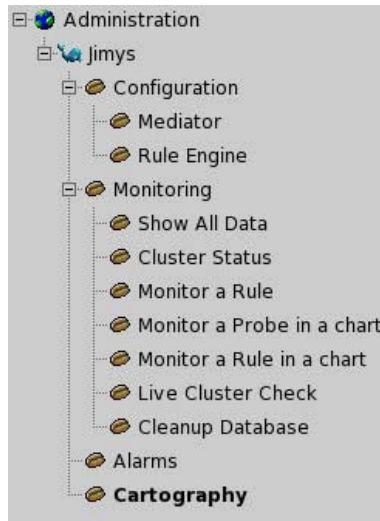
Time Stamp	Result Message	Level	Action
1123834994751	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123834999738	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel
1123835000329	fidji.dc.frec.bull.fr : CPU >=10 and hitsCount>=2	alert	Cancel

The level of the alert is set to “cancelled” and not shown any more on this page. To view the alert detail, it is necessary to consult Monitoring > Show All Data


- Show Alarm Detail

Alarms		
Show Alarms Show Alarm Detail		
ProbeIdentifier	Attribute Name	Attribute Value
fidji.dc.frec.bull.fr;domain;instance;probe;ujf;mbean=cpu	CPUload	31
fidji.dc.frec.bull.fr;jonas;jonas;probe;ujf;mbean=jonasjmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	hitsCount	2551
fidji.dc.frec.bull.fr;jonas;jonas;probe;ujf;mbean=jonasjmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	spareNotFoundEntries	500
fidji.dc.frec.bull.fr;jonas;jonas;probe;ujf;mbean=jonasjmx;jonas.type=Cache,host=localhost,path=/jonasAdmin	accessCount	2854

7.6 Cartography



To show the cartography, it is necessary to be connected to at least one pump.



Cartography

Cartography

- Your are not connected to a node, you need to be connected to at least to 1 pump

Select a host to get his cartography

No Cartography for host found


Cartography

Cartography

Select a host to get his cartography

☐ samoa.dc.frec.bull.fr

☒ fidji.dc.frec.bull.fr

No Cartography for host found

Choosing a host and selecting “Get”, the cartography of that host is shown to the user

Cartography

Cartography

Select a host to get his cartography

samoa.dc.frec.bull.fr

fidji.dc.frec.bull.fr

Get

Identifier	Key	Value
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cartography_Controller_PCI device 1166	Type	Host bridge
	Controllerid	2
	Name	PCI device 1166
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cartography_Controller_ServerWorks CNB20HE Host Bridge	Type	Host bridge
	Controllerid	2
	Name	ServerWorks CNB20HE Host Bridge
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cartography_Controller_ServerWorks CSB5 South Bridge	Type	ISA bridge
	Controllerid	1
	Name	ServerWorks CSB5 South Bridge
fidji.dc.frec.bull.fr;domain;instance;probe;ujf:mbean=cartography_Controller_ServerWorks GCLE Host Bridge	Type	Host bridge
	Controllerid	1
	Name	ServerWorks GCLE Host Bridge