Type of Monitors and Their Usage

There are two types of monitors: performance monitors and non-performance monitors.
The performance monitors have history and alerting mechanisms, which includes thresholds (used to compare the actual monitored value with a predefined value, and according to the result, to map a color to the monitor and create alerts). The performance monitors are: availability-monitor, integer-monitor, long-monitor, frequency-monitor, quality-rate-monitor, and duration-monitor.
The non-performance monitors do not have history and beside the state monitor, no alert mechanisms. The non-performance monitors are: text-monitor, state-monitor, table-monitor, version-monitor, and configuration-monitor.

1) Non-performance Monitors

text-monitor

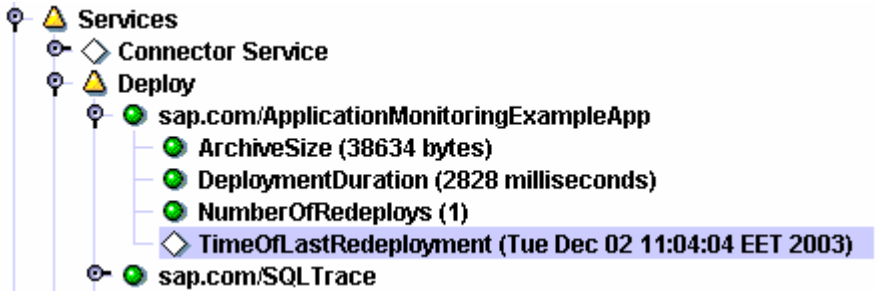
Usage: Monitors a String.
This monitor just look to the String value and shows it in the monitor tree.

Example:

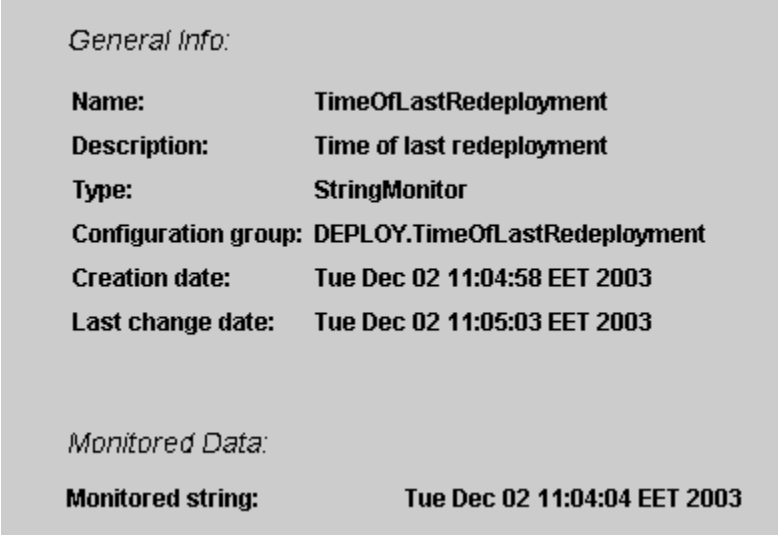
```
<text-monitor name="TimeOfLastRedeployment" configuration-
group="DEPLOY.TimeOfLastRedeployment">
        <monitored-resource name="deploy" type="SERVICE"/>
        <text-attribute-mapping>
                <text-attribute>
                        <observed-resource-attribute name="TimeOfLastRedeployment"/>
                </text-attribute>
        </text-attribute-mapping>
</text-monitor>
```

where method returning the monitored String is :
public String getTimeOfLastRedeployment();
(in this example this is a String representation of the time of the last redeployment)

representation in the tree:



representation in the viewer:



state-monitor

Usage: Monitors a String but with additional property of assigning colors (green, yellow, red) to the String values.

Must be defined a mapping from specific states to different colors (red, yellow, and green). The monitor compares the monitored states with the predefined ones, and according to the result, colors the state monitor nodes and potentially generates alerts.
For example state monitor can be used for the expiration state of some certificate.

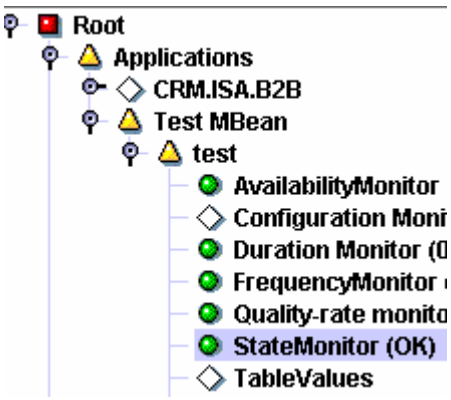
Example:

```
<state-monitor name="StateMonitor" configuration-group="StateMonitor">
        <monitored-resource name="TestMonitorMBean:j2eeType=SAP_TestMonitor_MBeans"
type="MBEAN"/>
        <state-attribute-mapping>
                <state-attribute>
                        <observed-resource-attribute name="TestState"/>
                </state-attribute>
        </state-attribute-mapping>
</state-monitor>
```

where the method returning the monitored state is :

```
public String getTestState();
```

representation in the tree:



representation in the viewer:

and the states for this example are :

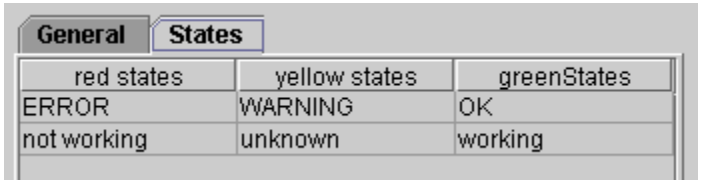


table-monitor

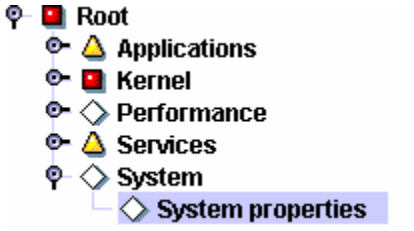
Usage: Monitors a table (each table element is a serializable object).

Example:
<table-monitor name="System properties" filename="SYSTEM.SYS_PROPS" configuration-group="SYSTEM.System_Props ">

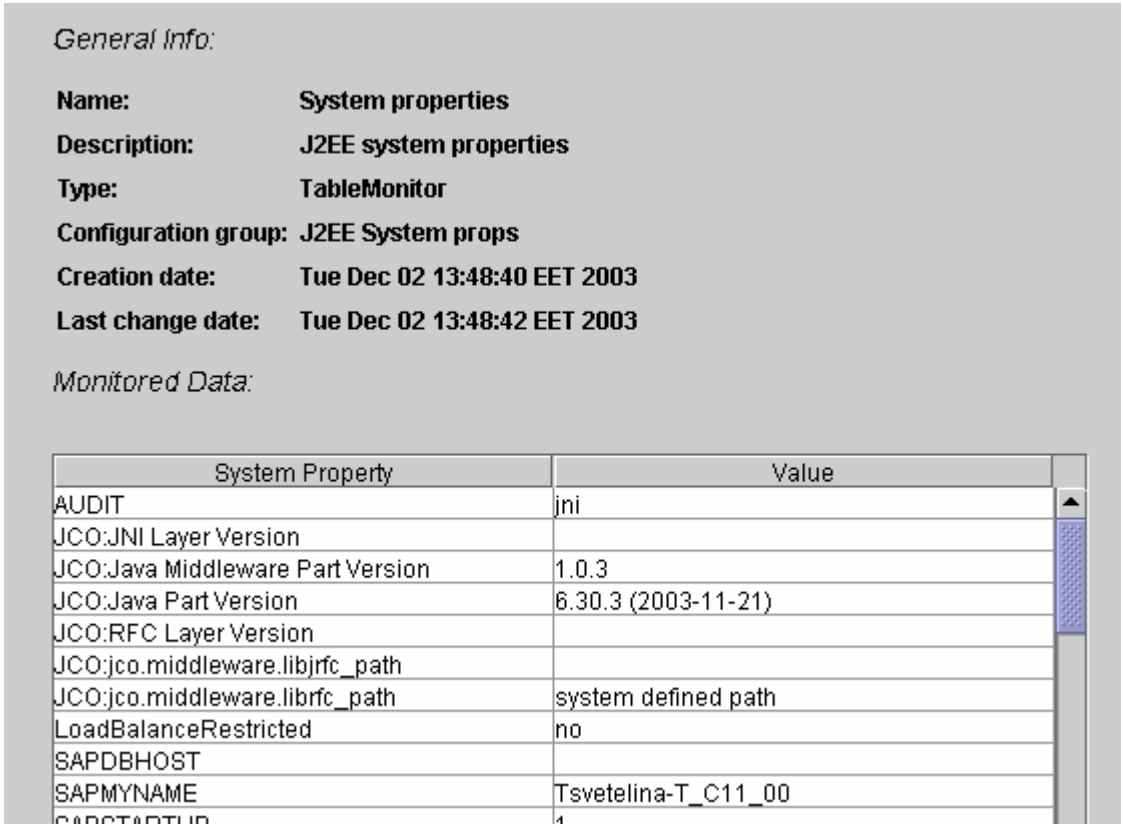
```
        <monitored-resource name="monitor" type="SERVICE"/>
                <table-attribute-mapping>
                        <table-entries-attribute>
                                <observed-resource-attribute name="SystemProperties"/>
                </table-attribute-mapping>
</table-monitor>
```

where method returning the monitored table is :
public Serializable[][] getSystemProperties ();

representation in the tree:



representation in the viewer:



version-monitor

Usage: Monitors VersionInfo, which is a version information of a component. The reported version consists of an application name, a major version, a minor version, a support

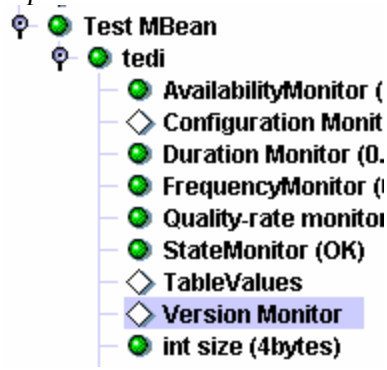
package number, a build time, a change list number, and an optional entry for additional information.

Example:

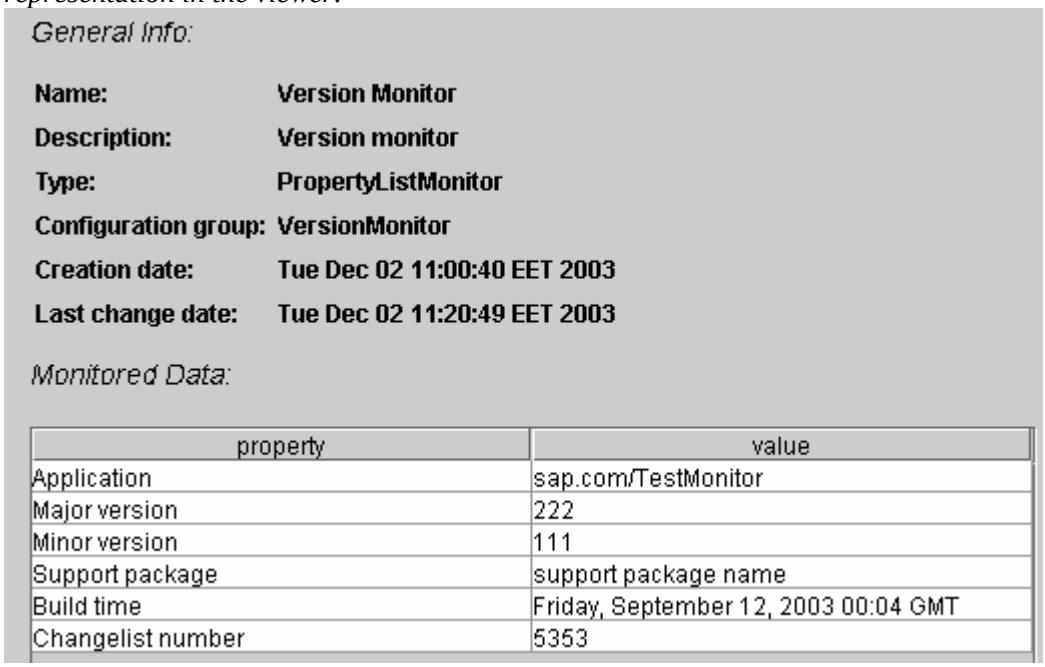
```
<version-monitor name="Version Monitor" configuration-group="TEST.VersionMonitor">
        <monitored-resource name="TestMonitorMBean:j2eeType=SAP_TestMonitor_MBeans"
type="MBEAN"/>
        <version-attribute-mapping>
                <version-attribute>
                        <observed-resource-attribute name="VersionParameters"/>
                </version-attribute>
        </version-attribute-mapping>
</version-monitor>
```

where the method returning the monitored version information is :
public com.sap.jmx.monitoring.api.VersionInfo getVersionParameters ();

representation in the tree:



representation in the viewer:



configuration-monitor

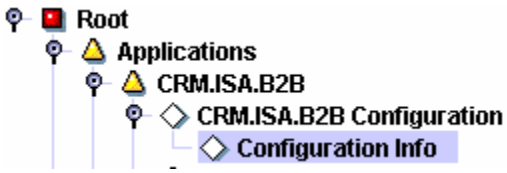
Usage: Monitors ConfigurationList - some configuration parameters of a particular component.

Example:

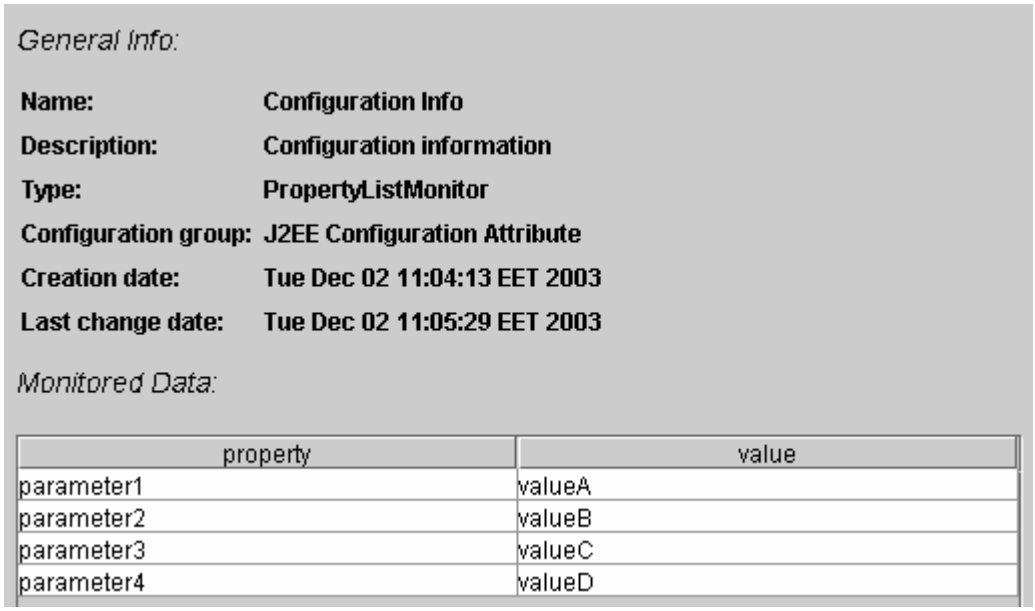
```
 <configuration-monitor name="Configuration Info" configuration-group="J2EE Configuration Attribute">
        <monitored-resource name="com.sap.jmx.monitoring.examples.MyConfigResourceMBean" type="APPLICATION"/>
        <configuration-attribute-mapping>
                <configuration-attribute>
                        <observed-resource-attribute name="ConfigurationParameters"/>
                </configuration-attribute>
        </configuration-attribute-mapping>
</configuration-monitor>
```

where the method returning the monitored configuration is :
public com.sap.jmx.monitoring.api.ConfigurationList getConfigurationParameters();

representation in the tree:



representation in the viewer:



2) Performance Monitors

availability-monitor

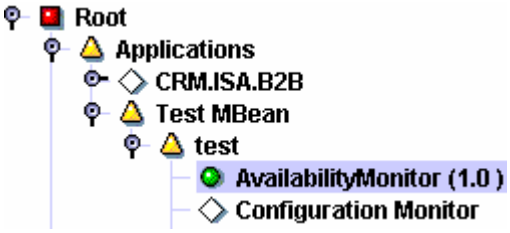
Usage: Monitors a boolean value e.g. used to monitor the availability of a resource. For example does

```
<availability-monitor name="AvailabilityMonitor" configuration-group="AvailabilityValue">
        <monitored-resource name="TestMonitorMBean:j2eeType=SAP_TestMonitor_MBeans"
type="MBEAN"/>
          <availability-attribute-mapping>
          <availability-attribute>
                  <observed-resource-attribute name="Availability"/>
           </availability-attribute>
          </availability-attribute-mapping>
  </availability-monitor>
```

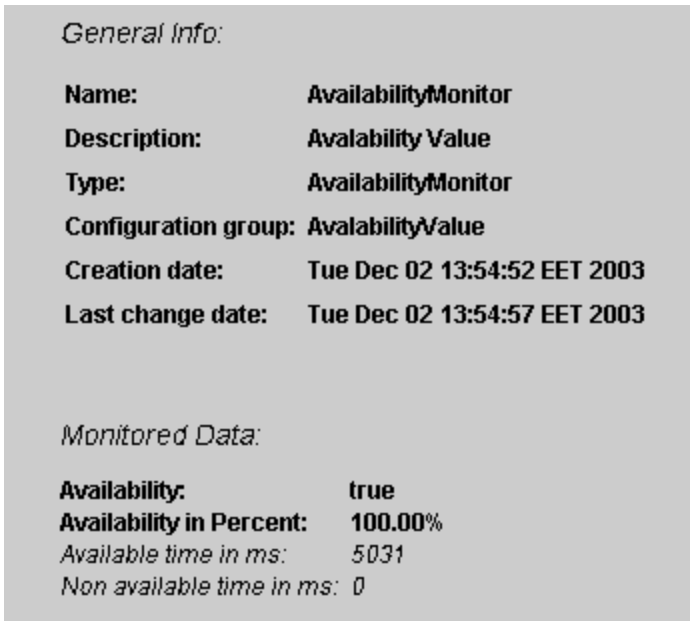
where the method returning the monitored value is

```
public boolean getAvailability ();
```

representation in the tree:



representation in the viewer:



integer-monitor

Usage: Monitors an int value. It can be for the number of threads, database connections, classloaders and so on.

Example:

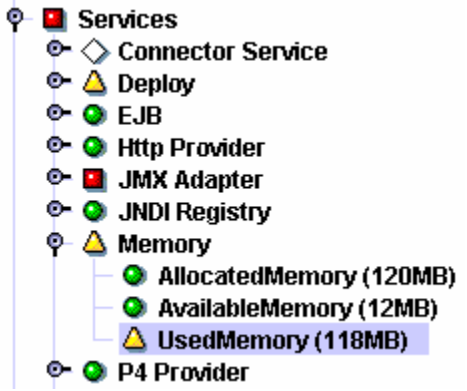
```
<integer-monitor name="UsedMemory" configuration-group="MEMORY.UsedMemory">
        <monitored-resource name="memory" type="SERVICE"/>
        <integer-attribute-mapping>
                <integer-attribute>
                        <observed-resource-attribute name="UsedMemory"/>
                </integer-attribute>
        </integer-attribute-mapping>
</integer-monitor>
```

where the method returning the monitored integer value is:

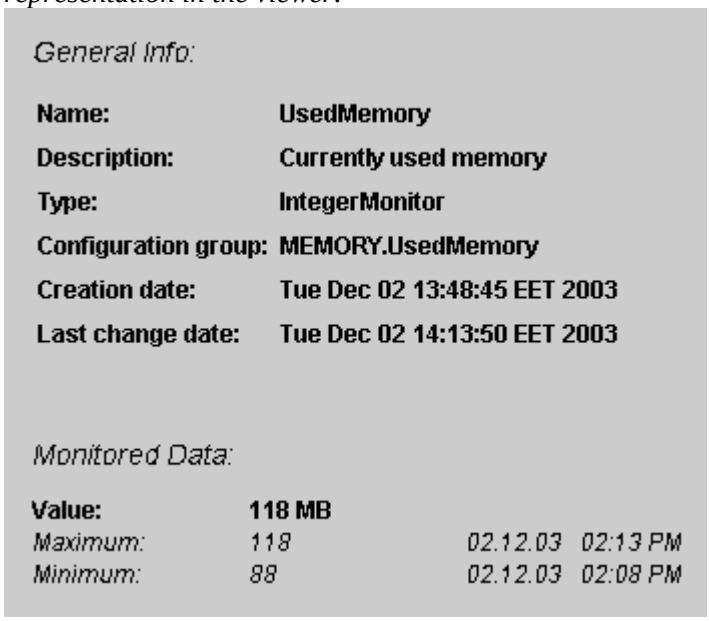
```
public int getUsedMemory();
```

(in this example this is the currently used memory in MB)

representation in the tree:



representation in the viewer:



long-monitor

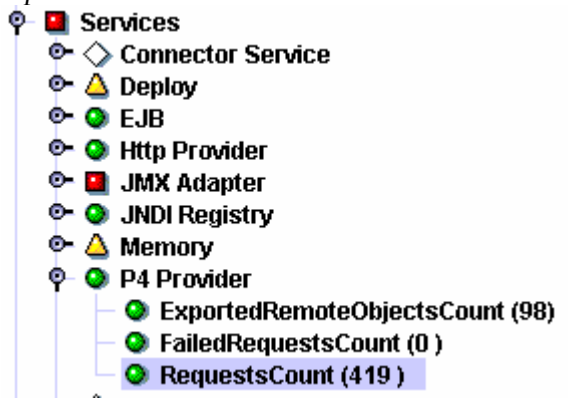
Usage: Monitors a long value. It can be used for

```
<long-monitor name="RequestsCount" configuration-group="P4.RequestsCount">
        <monitored-resource name="p4" type="SERVICE"/>
                <long-attribute-mapping>
                        <long-attribute>
                                <observed-resource-attribute name="RequestCount"/>
                        </long-attribute>
                </long-attribute-mapping>
</long-monitor>
```

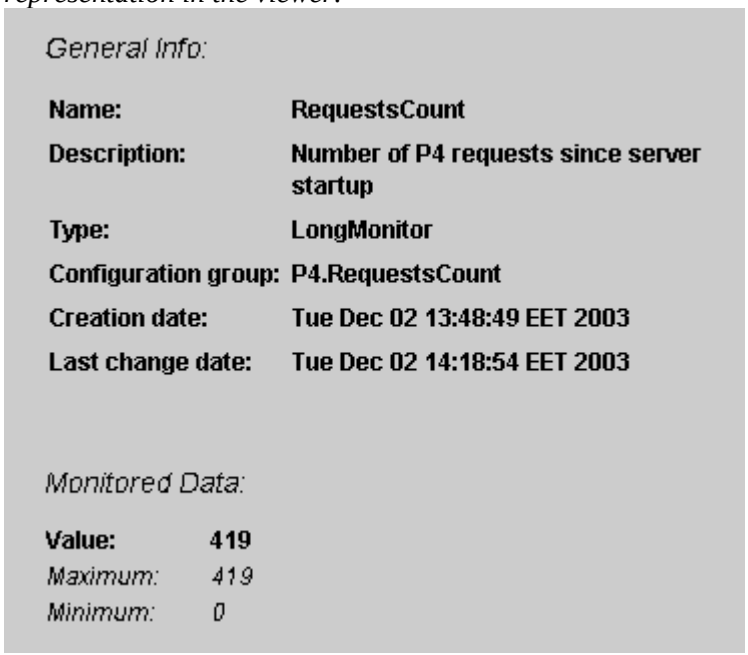
where the method returning the monitored long value is (in the example this is the number of P4 requests since server startup) :

```
public long getRequestCount();
```

representation in the tree:



representation in the viewer:



frequency-monitor

Usage: computes a frequency according to reported number of events given at specific times, by the rule:
frequency = reported number of events / time interval
For example if you want to know for some application the number of http request per second And there are 300 requests to the application for one minute, that means the the frequency is 5 requests/per second.

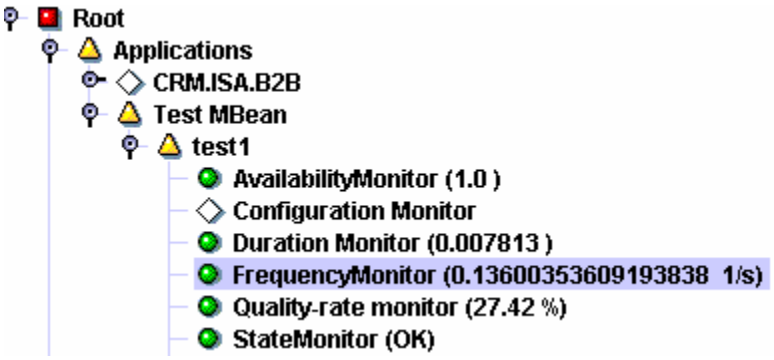
Example:

```
<frequency-monitor name="FrequencyMonitor" configuration-group="FrequencyMonitor">
        <monitored-resource
name="TestMonitorMBean:j2eeType=SAP_TestMonitor_MBeans" type="MBEAN"/>
        <frequency-attribute-mapping>
                <reported-events-attribute>
                        <observed-resource-attribute name="ReportedEvents"/>
                </reported-events-attribute>
        </frequency-attribute-mapping>
</frequency-monitor>
```

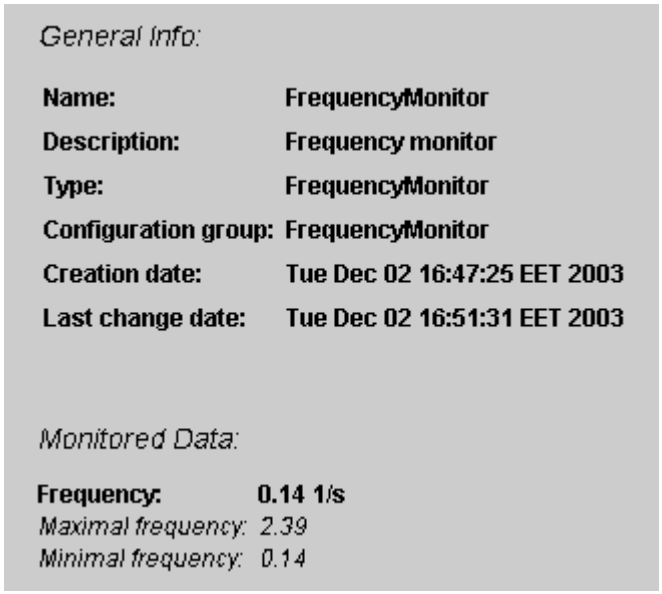
where the method returning the number of events is:

```
public int getReportedEvents();
```

representation in the tree:



representation in the viewer:



quality-rate-monitor

Usage: Computes an average (and actual) quality rate according to reported number of total tries and successful tries.

For example you have 1 successful try from a total amount of 10 tries. Then the average quality rate would be 10%.

Example :

Case 1)

```
<quality-rate-monitor name="Cache Hit Rate" configuration-group="ConfigMgr.CacheHitRate">
        <monitored-resource name="ConfigurationManager" type="MANAGER"/>
                <quality-rate-attribute-mapping>
                <total-tries-attribute>
                        <observed-resource-attribute name="AccNrOfRequests"/>
                </total-tries-attribute>
                <total-hits-attribute>
                        <observed-resource-attribute name="AccNrOfCacheRequests"/>
                </total-hits-attribute>
        </quality-rate-attribute-mapping>
</quality-rate-monitor>
```

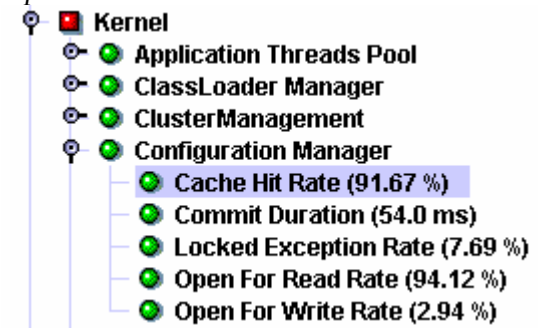
where the method returning the total tries from startup is
public long getAccNrOfRequests ();
and
where the method returning the total successful tries from startup
public long getAccNrOfCacheRequests ();

Case 2)

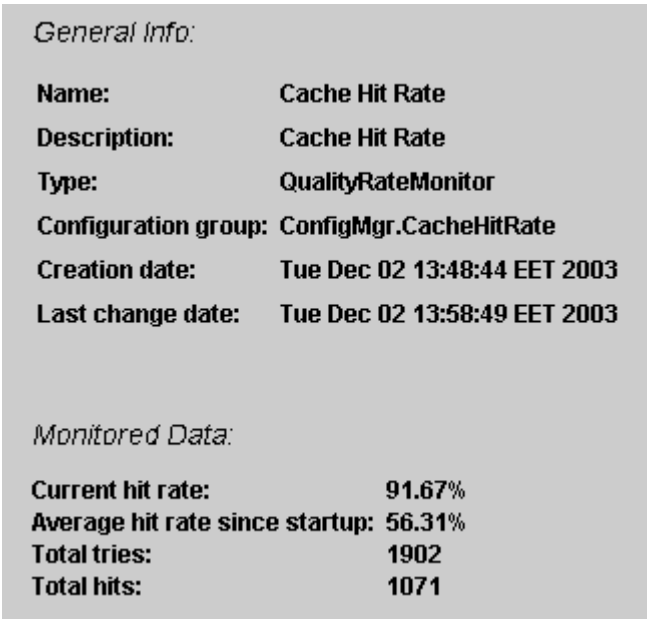
```
<quality-rate-monitor name="Cache Hit Rate" configuration-group="ConfigMgr.CacheHitRate">
        <monitored-resource name="ConfigurationManager" type="MANAGER"/>
                <quality-rate-attribute-mapping>
                <quality-rate-attribute>
                        <observed-resource-attribute name="QualityRate"/>
                </quality-rate-attribute>
        </quality-rate-attribute-mapping>
</quality-rate-monitor>
```

where the method returning the necessary data is
public com.sap.jmx.monitoring.api.QualityRateValue getQualityRate();

representation in the tree:



representation in the viewer:



duration-monitor

Usage: Compute the average time for a process. Defines a monitor for measuring the time for the http responses, session duration, etc.

If N and T are respectively the total number of responses and the total response time, the values under the last measurement node are calculated as follows:

number of requests = Ni - Ni-1

average time = Ti - Ti-1 / Ni - Ni-1 or 0 if Ni = Ni-1.

Example:

Case 1)

```
<duration-monitor name="Commit Duration" configuration-
group="ConfigMgr.CommitDuration">
        <monitored-resource name="ConfigurationManager" type="MANAGER"/>
        <duration-attribute-mapping>
                <total-number-attribute>
                        <observed-resource-attribute name="AccNrOfCommits"/>
                </total-number-attribute>
                <total-time-attribute>
                        <observed-resource-attribute name="AccCommitTime"/>
                </total-time-attribute>
        </duration-attribute-mapping>
</duration-monitor>
```

where the method returning the total number of tries
(in the example - Returns the accumulated number of commit-requests since the startup of the server)

```
public long getAccNrOfCommits();
```

and
the method returning the total time is :
(in the example - Returns the accumulated time of processing of all commit-requests since the startup of the server in milliseconds)

```
public long getAccCommitTime();
```

Case 2)

```
<duration-monitor name="Commit Duration" configuration-
group="ConfigMgr.CommitDuration">
        <monitored-resource name="ConfigurationManager" type="MANAGER"/>
        <duration-attribute-mapping>
                <duration-attribute>
                        <observed-resource-attribute name="DurationValue "/>
                </duration-attribute>
        </duration-attribute-mapping>
</duration-monitor>
```

where the method returning the necessary data is :

```
public com.sap.jmx.monitoring.api.DurationValue getDurationValue();
```

representation in the tree:

representation in the viewer:

General Info:

Name:	**Commit Duration**
Description:	**Commit Duration**
Type:	**DurationMonitor**
Configuration group:	**ConfigMgr.CommitDuration**
Creation date:	**Tue Dec 02 13:48:44 EET 2003**
Last change date:	**Tue Dec 02 13:58:49 EET 2003**

Monitored Data:

current average time:	**54.0ms**
average time since startup:	**169.48ms**
total number:	**60**
total time:	**10169**

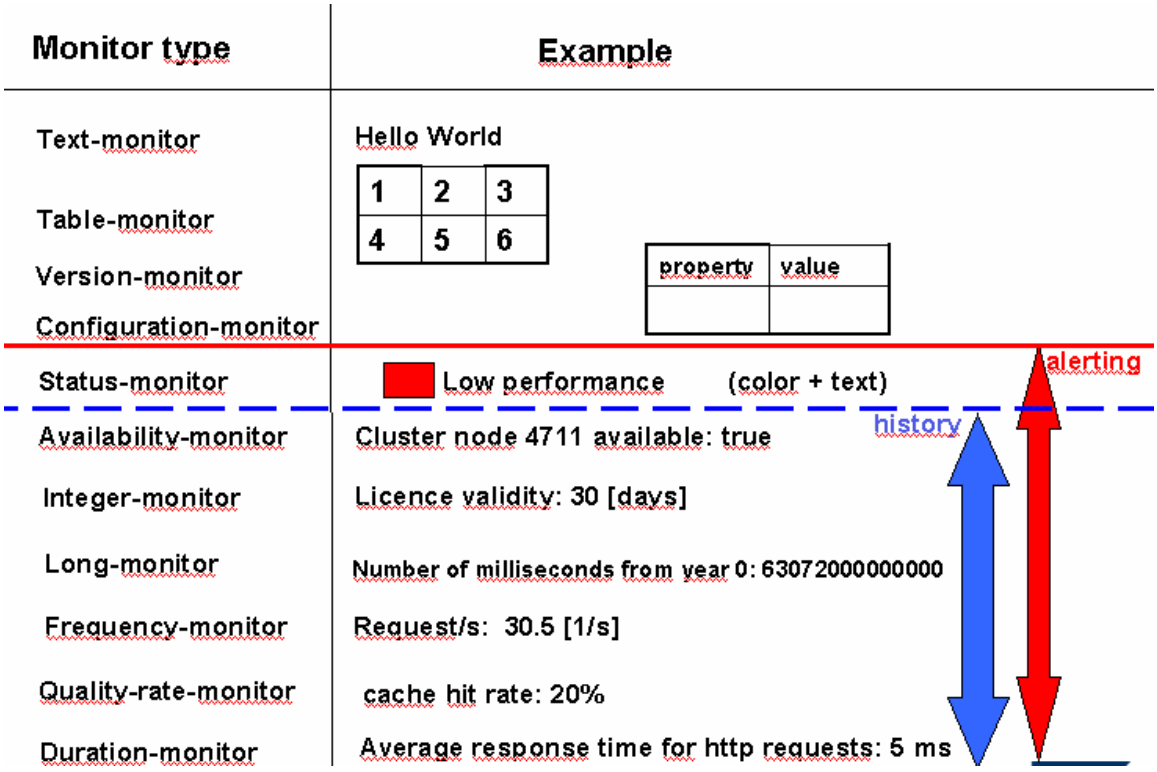


Fig. A slide from the presentation of Joerg Weller , which can be found here: http://bis.wdf.sap-ag.de:1080/general/docs/quickies/2003-11-19/ppt/ok_NEU_JMX_Monitoring.ppt