| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV PACKAGE**](http://docs.google.com/javax/management/modelmbean/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/management/openmbean/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/management/monitor/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package javax.management.monitor

Provides the definition of the monitor classes.

**See:**

[**Description**](#3znysh7)

| **Interface Summary** | |
| --- | --- |
| [**CounterMonitorMBean**](http://docs.google.com/javax/management/monitor/CounterMonitorMBean.html) | Exposes the remote management interface of the counter monitor MBean. |
| [**GaugeMonitorMBean**](http://docs.google.com/javax/management/monitor/GaugeMonitorMBean.html) | Exposes the remote management interface of the gauge monitor MBean. |
| [**MonitorMBean**](http://docs.google.com/javax/management/monitor/MonitorMBean.html) | Exposes the remote management interface of monitor MBeans. |
| [**StringMonitorMBean**](http://docs.google.com/javax/management/monitor/StringMonitorMBean.html) | Exposes the remote management interface of the string monitor MBean. |

| **Class Summary** | |
| --- | --- |
| [**CounterMonitor**](http://docs.google.com/javax/management/monitor/CounterMonitor.html) | Defines a monitor MBean designed to observe the values of a counter attribute. |
| [**GaugeMonitor**](http://docs.google.com/javax/management/monitor/GaugeMonitor.html) | Defines a monitor MBean designed to observe the values of a gauge attribute. |
| [**Monitor**](http://docs.google.com/javax/management/monitor/Monitor.html) | Defines the part common to all monitor MBeans. |
| [**MonitorNotification**](http://docs.google.com/javax/management/monitor/MonitorNotification.html) | Provides definitions of the notifications sent by monitor MBeans. |
| [**StringMonitor**](http://docs.google.com/javax/management/monitor/StringMonitor.html) | Defines a monitor MBean designed to observe the values of a string attribute. |

| **Exception Summary** | |
| --- | --- |
| [**MonitorSettingException**](http://docs.google.com/javax/management/monitor/MonitorSettingException.html) | Exception thrown by the monitor when a monitor setting becomes invalid while the monitor is running. |

## Package javax.management.monitor Description

Provides the definition of the monitor classes. A Monitor is an MBean that periodically observes the value of an attribute in one or more other MBeans. If the attribute meets a certain condition, the Monitor emits a [MonitorNotification](http://docs.google.com/javax/management/monitor/MonitorNotification.html). When the monitor MBean periodically calls [getAttribute](http://docs.google.com/javax/management/MBeanServer.html#getAttribute(javax.management.ObjectName,%20java.lang.String)) to retrieve the value of the attribute being monitored it does so within the access control context of the [Monitor.start()](http://docs.google.com/javax/management/monitor/Monitor.html#start()) caller.

The value being monitored can be a simple value contained within a complex type. For example, the [MemoryMXBean](http://docs.google.com/java/lang/management/MemoryMXBean.html) defined in java.lang.management has an attribute HeapMemoryUsage of type [MemoryUsage](http://docs.google.com/java/lang/management/MemoryUsage.html). To monitor the amount of *used* memory, described by the used property of MemoryUsage, you could monitor "HeapMemoryUsage.used". That string would be the argument to [setObservedAttribute](http://docs.google.com/javax/management/monitor/MonitorMBean.html#setObservedAttribute(java.lang.String)).

The rules used to interpret an ObservedAttribute like "HeapMemoryUsage.used" are as follows. Suppose the string is *A.e* (so *A* would be "HeapMemoryUsage" and *e* would be "used" in the example).

First the value of the attribute *A* is obtained. Call it *v*. A value *x* is extracted from *v* as follows:

* If *v* is a [CompositeData](http://docs.google.com/javax/management/openmbean/CompositeData.html) and if *v*.[get](http://docs.google.com/javax/management/openmbean/CompositeData.html#get(java.lang.String))(*e*) returns a value then *x* is that value.
* If *v* is an array and *e* is the string "length" then *x* is the length of the array.
* If the above rules do not produce a value, and if [Introspector.getBeanInfo](http://docs.google.com/java/beans/Introspector.html#getBeanInfo(java.lang.Class)) for the class of *v* (*v*.getClass()) contains a [PropertyDescriptor](http://docs.google.com/java/beans/PropertyDescriptor.html) with the name *e*, then *x* is the result of calling the property's [read method](http://docs.google.com/java/beans/PropertyDescriptor.html#getReadMethod()) on *v*.

The third rule means for example that if the attribute HeapMemoryUsage is a MemoryUsage, monitoring "HeapMemoryUsage.used" will obtain the observed value by calling MemoryUsage.getUsed().

If the ObservedAttribute contains more than one period, for example "ConnectionPool.connectionStats.length", then the above rules are applied iteratively. Here, *v* would initially be the value of the attribute ConnectionPool, and *x* would be derived by applying the above rules with *e* equal to "connectionStats". Then *v* would be set to this *x* and a new *x* derived by applying the rules again with *e* equal to "length".

Although it is recommended that attribute names be valid Java identifiers, it is possible for an attribute to be called HeapMemoryUsage.used. This means that an ObservedAttribute that is HeapMemoryUsage.used could mean that the value to observe is either an attribute of that name, or the property used within an attribute called HeapMemoryUsage. So for compatibility reasons, when the ObservedAttribute contains a period (.), the monitor will check whether an attribute exists whose name is the full ObservedAttribute string (HeapMemoryUsage.used in the example). It does this by calling [getMBeanInfo](http://docs.google.com/javax/management/MBeanServer.html#getMBeanInfo(javax.management.ObjectName)) for the observed MBean and looking for a contained [MBeanAttributeInfo](http://docs.google.com/javax/management/MBeanAttributeInfo.html) with the given name. If one is found, then that is what is monitored. If more than one MBean is being observed, the behavior is unspecified if some of them have a HeapMemoryUsage.used attribute and others do not. An implementation may therefore call getMBeanInfo on just one of the MBeans in this case. The behavior is also unspecified if the result of the check changes while the monitor is active.

The exact behavior of monitors is detailed in the [JMX Specification](#1t3h5sf). What follows is a summary.

There are three kinds of Monitors:

* A [CounterMonitor](http://docs.google.com/javax/management/monitor/CounterMonitor.html) observes attributes of integer type. The attributes are assumed to be non-negative, and monotonically increasing except for a possible *roll-over* at a specified *modulus*. Each observed attribute has an associated *threshold* value. A notification is sent when the attribute exceeds its threshold.  
  An *offset* value can be specified. When an observed value exceeds its threshold, the threshold is incremented by the offset, or by a multiple of the offset sufficient to make the threshold greater than the new observed value.  
  A CounterMonitor can operate in *difference mode*. In this mode, the value compared against the threshold is the difference between two successive observations of an attribute.
* A [GaugeMonitor](http://docs.google.com/javax/management/monitor/GaugeMonitor.html) observes attributes of numerical type. Each observed attribute has an associated *high threshold* and *low threshold*.  
  When an observed attribute crosses the high threshold, if the *notify high* flag is true, then a notification is sent. Subsequent crossings of the high threshold value will not trigger further notifications until the gauge value becomes less than or equal to the low threshold.  
  When an observed attribute crosses the low threshold, if the *notify low* flag is true, then a notification is sent. Subsequent crossings of the low threshold value will not trigger further notifications until the gauge value becomes greater than or equal to the high threshold.  
  Typically, only one of the notify high and notify low flags is set. The other threshold is used to provide a *hysteresis* mechanism to avoid the repeated triggering of notifications when an attribute makes small oscillations around the threshold value.  
  A GaugeMonitor can operate in *difference mode*. In this mode, the value compared against the high and low thresholds is the difference between two successive observations of an attribute.
* A [StringMonitor](http://docs.google.com/javax/management/monitor/StringMonitor.html) observes attributes of type String. A notification is sent when an observed attribute becomes equal and/or not equal to a given string.

**Since:** 1.5 **See Also:** [Java SE 6 Platform documentation on JMX technology](http://docs.google.com/technotes/guides/jmx/), in particular the  [JMX Specification, version 1.4(pdf).](http://docs.google.com/technotes/guides/jmx/JMX_1_4_specification.pdf)

| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV PACKAGE**](http://docs.google.com/javax/management/modelmbean/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/management/openmbean/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/management/monitor/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).