Build a Monitoring Extension Using Scripts

You can write a monitoring extension script (formerly known as a custom monitor) to add custom metrics to the metric set that AppDynamics already collects and reports to the Controller. Your script reports the custom metrics every minute to the standalone Machine Agent. The standalone Machine Agent passes these metrics on to the Controller.

This topic describes the steps for adding custom metrics using a shell script and includes an example.

Supported Operations on the Metrics

The Aggregation Qualifier

- The **aggregator** qualifier specifies how the standalone Machine Agent aggregates the values reported during a one-minute period.
- This value is an enumerated type. Valid values are:

Aggregator	Description
AVERAGE	Average of all reported values in that minute. The default operation.
SUM	Sum of all reported values in that minute. This operation behaves like a counter.
OBSERVATION	Last reported value in the minute. If no value is reported in that minute, the value from the last time it was reported is used.

The Time Roll Up Qualifier

- The **time-rollup** qualifier specifies how the Controller rolls up the values when it converts from one-minute granularity tables to 10-minute granularity and 60-minute granularity tables over time.
- The value is an enumerated type. Valid values are:

Roll up Strategy	Description
AVERAGE	Average of all one-minute data points when adding it to the 10-minute or 60-minute granularity table.
SUM	Sum of all one-minute data points when adding it to the 10-minute or 60-minute granularity table.
CURRENT	Last reported one-minute data point in that 10-minute or 60-minute interval.

The Cluster Roll Up Qualifier

- The **cluster-rollup** qualifier specifies how the Controller aggregates metric values in a tier (a cluster of nodes).
- The value is an enumerated type. Valid values are:

Roll up Strategy	Description
INDIVIDUAL	Aggregates the metric value by averaging the metric values across each node in the tier.
COLLECTIVE	Aggregates the metric value by adding up the metric values for all the nodes in the tier.

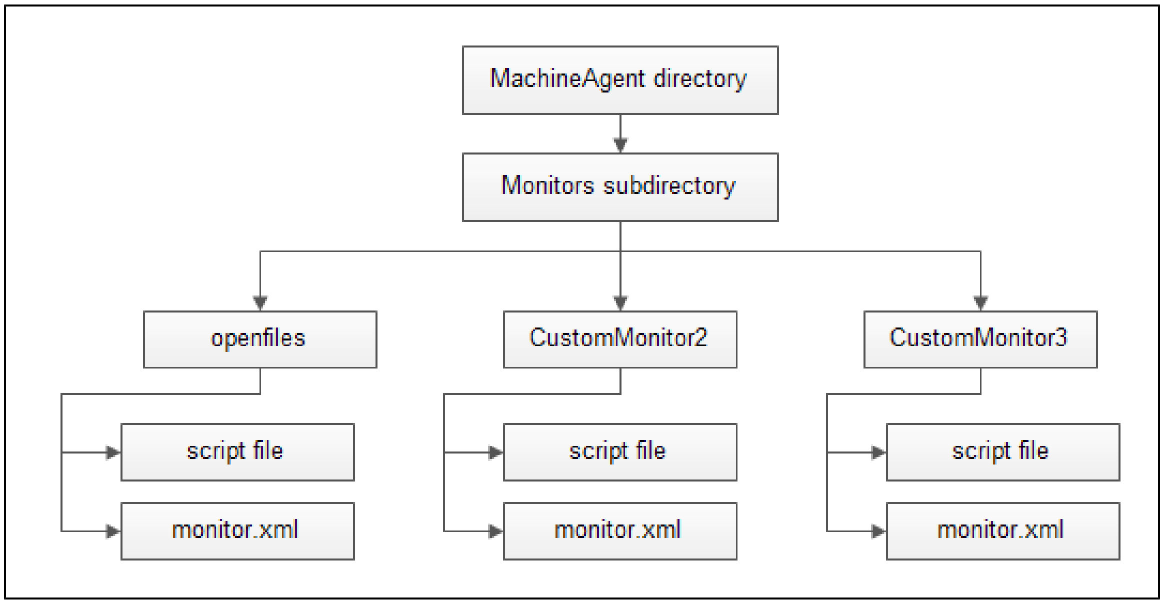
For example, if a tier has two nodes, Node A and Node B, and Node A has 3 errors per minute and Node B has 7 errors per minute, the INDIVIDUAL qualifier reports a value of 5 errors per minute and and COLLECTIVE qualifier reports 10 errors per minute. INDIVIDUAL is appropriate for metrics such as % CPU Busy where you want the value for each node. COLLECTIVE is appropriate for metrics such as Number of Calls where you want a value for the entire tier.

Adding a Monitoring Extension

1. Create a directory under the Machine Agent monitors directory

The /monitors directory in the Machine Agent installation directory is the repository for all monitoring extensions. For each new extension, create a sub-directory under the /monitors directory.

For example, to create a monitoring extension that monitors open files in the JVM, create a sub-directory named "openfiles" under the <Machine_Agent_installation/monitors> directory.



2. Create the script file

A script writes data to STDOUT. The Machine Agent parses STDOUT and sends information to the Controller every minute. Use the following instructions to create the script file.

1. Specify a name-value pair for the metrics.
Each metric has a name-value pair that is converted to a java 'long' value. A typical metric entry in the script file has the following structure:

```
name=<metric name>,value=<long value>
```

Use the following format:

	Format
Standard Form	Hardware Resources| Instrument Name=Instrument Value

Fully Qualified Form	Hardware Resources| <metric name>,value=<long value>

2. Define the category of the metric, for example:

- Infrastructure (for the default hardware metrics, see Monitor Hardware)
- JVM (for the default metrics, see Monitor JVMs)
- Custom Metrics
  You can use the "|" separator to introduce further hierarchy in the metric name similar to the following:

```
Hardware Resources|Disks|Total Disk Usage %
Hardware Resources|Disks|Disk 1|Current Disk Usage %
Custom Metrics|MySQL|Avg Query Time
Custom Metrics|Apache|Avg Wait Time
```

3. To monitor multiple metrics in the same script file, print a different line for each one. For example:

```
name=Hardware Resources|Disks|Total Disk Usage %, value=23
name=Hardware Resources|Disks|Disk 1|Current Disk Usage %, value=56
name=Custom Metrics|MySQL|Avg Query Time, value=500
name=Custom Metrics|Apache|Avg Wait Time, value=$count
```

3. Copy the script file to the directory created in Step 1

Ensure that the Agent process has execute permissions not only for the script file but also for the contents of the file.

4. Create the monitor.xml file

Follow the steps listed below to create your monitor.xml file:

1. For each custom monitor create a monitor.xml file.

The monitor.xml file executes the script file created in Step 2. You can edit the following sample file to create your file.

```
<monitor>
    <name>HardwareMonitor</name>
       <type>managed</type>
        <description>Monitors system resources - CPU, Memory, Network I/O, and Disk
I/O.</description>
        <monitor-configuration>     </monitor-configuration>
        <monitor-run-task>
           <!--  Edit execution-style as needed. -->
           <execution-style>periodic</execution-style>
           <name>Run</name>
           <type>executable</type>
           <task-arguments></task-arguments>
            <executable-task>
                <type>file</type>
                <!--  Use only one file element per os-type. -->
                 <file os-type="linux">linux-stat.sh</file>
                 <file os-type="mac">macos-stat.sh</file>
                 <file os-type="windows">windows-stat.bat</file>
                 <file os-type="solaris">solaris-stat.sh</file>
                 <file os-type="sunos">solaris-stat.sh</file>
                 <file os-type="aix">aix-stat.sh</file>
           </executable-task>
         </monitor-run-task>
</monitor>
```

⚠ The os-type attribute is optional for the executable-task file element when only one os-type is specified. One monitor.xml file executes one script per os-type.

2. Select the execution style from one of the following:

Execution Style	Description	Example

Continuous	Choose "Continuous", if you want data collection averaged over time.
	(For example: Averaging CPU over minute).
	This ensures that the script keeps running until the machine agent process is terminated.(!) For the monitor to be declared as 'continuous', the script should also run in an infinite loop.
		while [1]; do
		... the actual script goes here ...
		sleep 60
		done

Periodic	Choose periodic, if you want data to be reported from system performance counters periodically.	The periodic task runs every minute by default. Use following parameter in the XML file, if you want the periodic task to run with any other frequency: <monitor-run-task> element. <execution-frequency-in-seconds>120</execution-frequency-in-seconds> For all the other frequency settings, the data is aggregated.

3. Add the name of this script file to the <file> element in the monitor.xml file. Be sure to use the correct os-type attribute. The os-type value should match the value returned from calling System.getProperty("os.name"). See http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/System.html#getProperties%28%29

```
<file os-type="your-os-type">{script file name}</file>
```

You can use either the relative or absolute path of the script.

5. Copy the monitor.xml file to the directory created in Step 1

6. Restart the standalone Machine Agent

After restarting the standalone Machine Agent, you should see following message in your log file:

```
Executing script [<script_name>] on the console to make sure your changes work with the
machine agent.
```

7. Verify execution of the monitoring extension script

To verify the execution of extension, wait for at least one minute and check the metric data in the Metric Browser.

You can now create alerts based on any of these metrics.

Example: Create a monitoring extension for open files

This section provides instructions to create a custom monitor for monitoring all the open files for JVMs.

1. Create a new directory in the custom monitor repository.

2. Create the script file.

This is a sample script. Modify this script for the specific process name (for example: Author, Publish, and so on).

```
lookfor="<process name 1>"
pid=`ps aux | grep "$lookfor" | grep -v grep | tr -s " " | cut -f2 -d' '`
count1=`lsof -p $pid | wc -l | xargs`

lookfor="<process name 2>"
pid=`ps aux | grep "$lookfor" | grep -v grep | tr -s " " | cut -f2 -d' '`
count2=`lsof -p $pid | wc -l | xargs`

echo "name=JVM|Files|<process name 1>,value="$count1
echo "name=JVM|Files|<process name 2>,value="$count2
```

3. Create the monitor.xml and point this XML file to the script file created in step 2.

```
<monitor>
    <name>MyMonitors</name>
    <type>managed</type>
    <description>Monitor open file count </description>
    <monitor-configuration>
    </monitor-configuration>
    <monitor-run-task>
        <execution-style>continuous</execution-style>
        <name>Run</name>
        <type>executable</type>
        <task-arguments>
        </task-arguments>
        <executable-task>
            <type>file</type>
            <file>openfilecount.sh</file>
        </executable-task>
    </monitor-run-task>
</monitor>
```

Learn More

- Administer Machine Agents
- Notification Actions
- Machine Agent Install and Admin FAQ
- Supported Environments and Versions