Server Monitoring

AppDynamics Application Intelligence Platform
Version 4.2.x
Server Monitoring

On this page:
- Feature Comparison
- Server Monitoring Role-Base Access Control
- Server Monitoring User Interface
- Watch the Video

Related pages:
- Enable Server Monitoring
- Monitor Your Servers using Server Monitoring
- Server Monitoring Metrics Reference

Server Monitoring uses the Standalone Machine Agent to provide extended hardware metrics and Service Availability Monitoring. Server Monitoring includes additional windows in the Controller UI that enable you to quickly see underlying infrastructure issues impacting your application performance. This means you can rapidly troubleshoot hardware performance problems that are impacting your applications.

Server Monitoring is not enabled by default and requires a separate license. Server Monitoring is currently available for Linux and Windows.

You need to install the Standalone Machine Agent to enable and use Server Monitoring and Service Availability features. See Install the Standalone Machine Agent.

Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Server Monitoring</th>
<th>Standalone Machine Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Platforms</td>
<td>Linux and Windows</td>
<td>Linux and Unix-like systems, Solaris, AIX, MacOS, and Windows</td>
</tr>
<tr>
<td>Extensions and Custom Metrics</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JVM Crash Guard</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Remediation Scripts</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Licensing</td>
<td>Server Infrastructure Monitoring</td>
<td>AppDynamics Machine Agent</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Yes</td>
<td>Limited to Servers list</td>
</tr>
<tr>
<td>Extended Hardware Metrics</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Service Availability Monitoring</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Server metrics in transaction snapshots</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Server KPIs in application flow map</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Health rules for extended hardware metrics</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Server Monitoring Role-Base Access Control
Server Monitoring provides two default roles and two related permissions:

- Server Monitoring Administrator
- Server Monitoring User
- View Server Monitoring permission
- Configure Server Monitoring (Service Availability) permission

For more details, see See Roles and Permissions.

Server Monitoring User Interface

The **Servers list** provides key performance metrics for all your servers on a single window.

The **Servers > Dashboard** shows key performance metrics for the selected machine, including the top 10 consumers of CPU and memory.

The **Servers > Volumes** window displays performance metrics for disks, partitions, and volumes for the selected machine.

The **Servers > Network** window shows network performance metrics for network interfaces on the selected machine.

The **Servers > Processes** window shows a configurable set of performance metrics for processes.

Using the Server Monitoring UI

The Server Monitoring user interface uses many of the same mechanisms that are common to the various windows of the Controller UI. The image below illustrates and briefly describes these mechanisms.

Watch the Video

Server Visibility Product Tour
Enable Server Monitoring

Server Monitoring features are available for Linux and Windows. You need a Server Monitoring license to enable and use Server Monitoring features. Server Monitoring is enabled by default on AppDynamics controllers. You must explicitly enable your Standalone Machine Agent to start sending the expanded set of Server Monitoring metrics.

Standalone Machine Agents can be installed on other supported operating systems to collect basic metrics and run extensions, however, the Server Monitoring features are not available.

Important Notes

- AppDynamics recommends that you use the latest available version of the Standalone Machine Agent for Server Monitoring.
- Monitored servers are considered stale 30 days after they go offline. They are purged from the Controller database when that time limit is reached.
- If you plan to enable Server Visibility on a Windows server where a .NET APM agent is installed, you must enable .NET Compatibility Mode on both the Controller and the Machine Agent. For more information, see .NET Compatibility Mode.
- For specific guidance on agent sizing and capacity considerations in your organization, please contact your AppDynamics account representative.

Enable Server Monitoring

This section assumes that you have already installed and configured the Standalone Machine Agent on a Windows or Linux machine as described in Install the Standalone Machine Agent.

To enable Server Monitoring

1. Set the "sim enabled" property to "true". For details on where to put configuration settings, see Where to Specify Machine Agent Configuration. Configure the value depending on your configuration choice:
   - **Element in controller-info.xml**: <sim-enabled>
   - **System Property**: -Dappdynamics.sim.enabled
   - **Environment Variable**: APPDYNAMICS_SIM_ENABLED
2. Restart the machine agent. See Start and Stop the Standalone Machine Agent.

Monitor Your Servers using Server Monitoring

This section describes the Server Monitoring windows in the Controller UI including: how to use the windows, what the metrics displayed mean, and a screenshot of the window:

- Discover Normal Server Activity
- Servers List
- Server Dashboard
- Server Volumes Metrics
- Server Network Metrics
- Server Processes
- View Server Metrics From Application Context

Discover Normal Server Activity

Server Monitoring automatically learns to detect performance anomalies using baselines that are specific to your server environments.

AppDynamics creates baselines by collecting metrics from your monitored servers over defined periods of time. This establishes what is normal for your environment and you can create health rules to alert you when metric deviations out of the normal range occur. You can also create your own baselines.

View Performance Metrics Compared to Baselines

You can compare performance metrics to their dynamic baselines on the Server dashboard.
You can visualize performance metrics and see how they deviate from expected behaviors established by the baseline on the Metric Browser for Server & Infrastructure Monitoring.
Access the Servers List from AppDynamics Home by clicking Servers in the top navigation bar. A scrollable list of all the servers you are monitoring appears. If a machine is not enabled for Server Monitoring, the Servers list shows only the machine name and no metric data is shown on the sparkline graphs or in the tabular display. The machines displayed on this list are the ones registered by the Standalone Machine Agent. Machines registered by the embedded .NET Machine Agent or machines registered by an App Agent (on a machine that does not have a Standalone Machine agent running) are not displayed in this list.

On the Servers list you can:

- See the complete list of machine agents reporting to this controller. The Name is taken from the value in `<machine-path>` in controller-info.xml. The last string after the last | (pipe) is used. For example, if the value is `<machine-path>Group | myName<machine-path>`, then the name will be "myName". For `<machine-path>JustName<machine-path>`, then the name will be "JustName".
- Filter the list using the dropdown to show all servers (the default) or just servers with Server Monitoring enabled or disabled.
- Use View Options to show optional fields, such as Trends and Host Id.
- See key performance metrics for your monitored servers at a glance.
- Change the time period of the metrics displayed.
- See an overview of the health of the servers, as determined by whether any health rules have been violated. See View Health Rule Status in the UI.
- Click on any point on a spark chart to see the percentage usage at that time.
- Sort the chart by any of the columns.
- Delete a machine from the Servers list. See Delete Machine.
- See additional details of servers that are enabled for Server Monitoring. Double-click a selected server to see server-specific dashboard, and details of the server's volumes, network, and processes.
- (*New in 4.2.13*) Disable automatic sorting of the Servers List table when it first loads. This option is useful if the Controller is monitoring hundreds or thousands of Standalone Machine Agents and the table takes a long time to load. This option also disables the ability to sort the table by any metric or metric trend column. To enable this option for an individual account, do the following:
  1. Log in to the Controller admin page (`<controller-hostname>/controller/admin.jsp`)
  2. From Accounts, open the account page of interest.
  3. Click + Add Property and enter:
     a. Property = SIM_DISABLE_MACHINE_SORTING
     b. value = true

### Server Details

<table>
<thead>
<tr>
<th>Name</th>
<th>Health</th>
<th>Disk Usage (%)</th>
<th>CPU (%)</th>
<th>Mem. (%)</th>
<th>Disk I/O (%)</th>
<th>Network I/O (%)</th>
<th>Server Mon. (Enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIN-INT194G483A</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Address Services</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Customer Survey</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>ECommerce Services</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>ECommerce Services</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Fulfillment</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Fulfillment-Client</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>
Metrics on the Servers List Window

The following describes the columns available for the Servers list window:

- **Hierarchy**: Specifies the root of the hierarchical path to the server. See the machine hierarchy property in this topic: Standalone Machine Agent Configuration Property Reference.
- **Health**: Indicates if server health rules have been violated on this machine.
- **Disk Usage % trend**: The percentage usage trend over time of storage space in use across all listed volumes, partitions, and disks.
- **CPU (%)**: The percentage of time the CPU was busy processing system or user requests.
- **Network I/O (%):** The percentage of memory used.
- **Memory Trend**: The trend over time of memory usage.
- **Disk I/O (%)**: The percentage of time spent performing read and write operations across one or more disks, volumes, or partitions.
- **Network Trend**: The trend over time of network usage.
- **Server Monitoring Enabled**: Indicates if Server Monitoring is enabled for this machine. Values=Yes, No, and Not Supported. Not Supported indicates that Server Monitoring is not available for that machine OS.
- **Icon**: representation of the machine's OS

For more details on the metrics shown here, see Server Monitoring Metrics Reference.

Delete Machine

Server Monitoring Administrator permission is required to delete machines.

You can delete machines from the Server list regardless of whether they are Server Monitoring enabled. You can select multiple machines for the delete action.

Deleting a machine from the Server list deletes the underlying machine metadata that displays on the AppDynamics Agents Admin page, but only if no APM nodes are currently associated with the same machine.

Deleting a machine that still has APM nodes currently associated with it removes the entry from the Servers list, but retains the underlying machine metadata. That information continues to be displayed on the AppDynamics Agents admin page.

Server Dashboard

Access the Server Dashboard from AppDynamics Home by clicking Servers from top navigation bar. Then select a server and double-click a server name. The Dashboard for the selected server appears.
On the Server Dashboard for the selected server you can:

- See charts of key performance metrics for the selected monitored servers, including server availability;
  - CPU, memory, and network usage percentages
  - server properties
  - disk, partition, and volume metrics
  - top 10 processes consuming CPU resources and memory
- Change the time period of the metrics displayed.
- See an assessment of the overall health of the server, as determined by whether any health rules have been violated. See View Health Rule Status in the UI.
- See the hierarchy or grouping of the server as specified in the controller-info.xml using the machine-path configuration property.
- Click on any point on a chart to see the metric value for that time.
- Sort the chart using any of the columns as a sorting key.

Interpreting the Server Identification Information and Metrics on the Server Dashboard Window

- **Host id**: This is an ID for the server that is unique to the AppDynamics Controller.
- **Health**: Shows the overall health of the server. Hover over the health icon for details; for example, the following appears when hovering over the exclamation mark ( ! ) in a red box, “There are Critical Health Rule Violations on this machine during the time range.”
- **Hierarchy**: This is an arbitrary hierarchy to group your servers together, specified by Machine Hierarchy Property.

The following describes the metrics available on the Server Dashboard:

**Availability**

- **Availability**: Percentage of time the server was available. The Standalone Machine Agent sends a heartbeat to the AppDynamics Controller once a minute to indicate the agent is alive.
- **time series chart**: Shows the server availability trend over time.
CPU

- **CPU Usage**: Average usage of CPU over the chosen time-range.
- **# of Cores**: Number of cores for the CPU.
- **time series chart**: Shows CPU busy percentage trend over time.

Memory

- **Installed**: Total amount of memory, free and used, on the server.
- **Memory Usage**: Percentage of memory used.
- **Swap Usage**: Swap usage tracks the swap file utilization. A swap file enables an operating system to use hard disk space to simulate extra memory. When the system runs low on memory, it swaps a section of RAM being used by an idle program onto the hard disk to free up memory for other executing programs. When the idle, swapped out program begins executing again, it is moved back to RAM, potentially displacing some other idle program. This causes a large amount of hard disk reading and writing that can slow down the computer considerably.
- **time series chart**: Shows the memory usage trend over time.

Network

- **Interfaces**: Number of network interfaces on the server.
- **Outgoing**: Kilobytes of data sent per second for all monitored network devices.
- **Incoming**: Kilobytes of data received per second for all monitored network devices.
- **time series chart**: Shows the network incoming and outgoing volume trend over time.

---

**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>os.name</code></td>
<td>Linux</td>
</tr>
<tr>
<td><code>os.arch</code></td>
<td>x86_64</td>
</tr>
<tr>
<td><code>os.version</code></td>
<td>4.1.10-17.31-amd64</td>
</tr>
<tr>
<td><code>machine-agent</code></td>
<td>3649060</td>
</tr>
<tr>
<td><code>java.vm.name</code></td>
<td>HotSpot(TM) 64-Bit Server VM</td>
</tr>
<tr>
<td><code>java.version</code></td>
<td>1.7.0_79</td>
</tr>
<tr>
<td><code>user.name</code></td>
<td>ANager</td>
</tr>
<tr>
<td><code>user.country</code></td>
<td>US</td>
</tr>
<tr>
<td><code>user.variant</code></td>
<td>unknown</td>
</tr>
<tr>
<td><code>Machine Agent v4.2.0 QA Build Date 2016-01-07</code></td>
<td></td>
</tr>
</tbody>
</table>

**Volumes**

<table>
<thead>
<tr>
<th>Volume</th>
<th>Total(MB)</th>
<th>Free(MB)</th>
<th>Usage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/hostname</td>
<td>201,457</td>
<td>188,314.1</td>
<td>6.4</td>
</tr>
<tr>
<td>/etc/resolve</td>
<td>201,457</td>
<td>188,314.1</td>
<td>6.4</td>
</tr>
<tr>
<td>/etc/hostss</td>
<td>201,457</td>
<td>188,314.1</td>
<td>6.4</td>
</tr>
<tr>
<td>/docker:20:1:2:3...</td>
<td>162,340</td>
<td>100,788.8</td>
<td>2</td>
</tr>
</tbody>
</table>

**Top 10 Processes Consuming CPU**

<table>
<thead>
<tr>
<th>Class</th>
<th>Count</th>
<th>CPU (%)</th>
<th>Memory (%)</th>
<th>PID</th>
<th>PPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>java</td>
<td>2</td>
<td>0.8</td>
<td>5.0</td>
<td>MULT1</td>
<td>MULT1</td>
</tr>
<tr>
<td>start-qs</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>startup.sh</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bash</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>175</td>
<td>102</td>
</tr>
<tr>
<td>start-appserver</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

**Top 10 Processes Consuming Memory**

<table>
<thead>
<tr>
<th>Class</th>
<th>Count</th>
<th>CPU (%)</th>
<th>Memory (%)</th>
<th>PID</th>
<th>PPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>java</td>
<td>2</td>
<td>0.8</td>
<td>5.0</td>
<td>MULT1</td>
<td>MULT1</td>
</tr>
<tr>
<td>start-qs</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>startup.sh</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bash</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>175</td>
<td>102</td>
</tr>
<tr>
<td>start-appserver</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

**Volumes**

- **Total**: Storage space, free and used, on the disk, partition or volume. For Linux systems, this does not include disk space reserved by the kernel.
- **Free**: Total storage space available. For Linux systems, this does not include disk space reserved by the kernel.
- **Usage(%)**: Percentage of storage space in use across each disk, partition and volume.
- **time series chart**: Shows the storage usage trend over time.

---

**Top 10 Processes Consuming CPU**

- **Count**: Number of processes in this class.
- **CPU (%)**: Percentage of CPU resources consumed by all processes in this class.
- **Memory (%)**: Percentage of memory consumed by all processes in the class.
- **PID**: Process ID.
Top 10 Processes Consuming Memory

- **Count**: The number of processes in this class consuming memory resources.
- **CPU (%)**: Percentage of CPU consumed by all processes in this class.
- **Memory (%)**: Percentage of memory consumed by all processes in this class.
- **PID**: Process ID.
- **PPID**: Parent Process ID

For more details, see [Server Monitoring Metrics Reference](#).

Server Volumes Metrics

<table>
<thead>
<tr>
<th>On this page:</th>
<th>Related pages:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Server Volumes</td>
<td>- Server Monitoring Metrics Reference</td>
</tr>
<tr>
<td>- Metrics on the Server Volumes Window</td>
<td></td>
</tr>
</tbody>
</table>

Access the Server Volumes data from AppDynamics Home top navigation bar by clicking **Servers**. On the Servers list double-click a server name, and click the **Volumes** tab.

Server Volumes

On the Servers Volume tab for the selected server you can:

- See in the list of volumes, percentage used and total storage space available on the disk, partition or volume.
- See disk usage and I/O utilization, rate, operations per second, and wait time.
- Change the time period of the metrics collected and displayed.
- Click on any point on a chart to see the metric value for that time.
Metrics on the Server Volumes Window

For each selected disk, partition or volume, the following describes the columns available for the Server Volumes window.

- **Disk Usage %**: The percentage of storage space in use.
- **I/O Utilization %**: The percentage of time spent performing read and write operations.
- **I/O Rate read**: The number of kilobytes per second of data read.
- **I/O Rate write**: The number of kilobytes per second of data write.
- **I/O Operations per sec read**: The number of read operations per second.
- **I/O Operations per sec write**: The number of write operations per second.
- **I/O Wait Time read**: The percentage of time the CPU was waiting for read operations to complete.
- **I/O Wait Time write**: The percentage of time the CPU was waiting for write operations to complete.

**Notes:**

- Volume space metrics on POSIX systems are obtained using the "df" command.
- On Linux, the space reserved for root is not counted in the available space.

For more details of the metrics shown here, see Server Monitoring Metrics Reference.

Server Network Metrics

From the AppDynamics Controller top navigation bar, click Servers. On the Servers list double-click a server name and click the Network tab.

Server Network Metrics

On the Server Network tab for the selected server you can:

- For each network interface, see the MAC, IPv4, and IPv6 address.
- See whether or not the network interface is:
  - enabled
  - functional, its operational state
  - equipped with an ethernet cable that is plugged in
  - operating in full or half-full duplex mode
- and the
  - maximum transmission unit (MTU) or size (in bytes) of the largest protocol data unit that the network interface can pass
  - speed of the ethernet connection in Mbit/sec
- View network throughput in kilobytes/sec and packet traffic.
- Change the time period of the metrics displayed.
- Hover over on any point on a chart to see the metric value for that time.

**Notes**

In some cases, the Standalone Machine Agent might be unable to determine the following:

- Network device speed for some devices and some Linux versions. In this case, no metrics will appear on this page.
- MTU for some Windows versions. In this case, the MTU will not appear on this page.
Metrics on the **Server** Network Window

The following describes the columns available for the Servers Network window.  

For the selected network interface:

- **Network Throughput received**: The volume of data received per second.
- **Network Throughput sent**: The volume of data sent per second.
- **Packets per sec incoming**: The number of data packets received per second.
- **Packets per sec outgoing**: The number of data packets sent per second.

For more details of the metrics shown here, see Server Monitoring Metrics Reference.

**Server Processes**

**On this page:**
- Access the Server Processes List
- Server Processes
- Metrics for Server Processes

**Related pages:**
- Server Monitoring Metrics Reference
- Enable Server Monitoring
- Configure Server Monitoring

You can configure which processes AppDynamics monitors, how they are grouped, number of processes to monitor, and how long they must be alive before monitoring them. For more information, see “Configure Process Monitoring” on Enable Server Monitoring.

**Access the Server Processes List**

From AppDynamics Home, click **Servers**. On the Servers list double-click a server name and click the **Processes** tab.
On the Server Processes tab for the selected server you can:

- View all the processes active during the selected time period. The processes are grouped by class as you specified in the `ServerMonitoring.yml` file.
- View the full command line that started this process by hovering over the process entry in the Command Line column.
- Expand a process class to see the processes associated with that class.
- Use View Options to configure which columns to display in the chart.
- Change the time period of the metrics displayed.
- Sort the chart using the columns as a sorting key. You can not sort on sparkline charts: CPU Trend and Memory Trend.
- See CPU and Memory usage trends at a glance

### Metrics for Server Processes

The following information is presented for each monitored class and process:

**Default Columns:**

- **Class:** The process class
- **Count:** The number of processes in this class.
- **Command Line:** The command that started the process.
- **State:** An icon represents the process state: sleeping, running, terminated, zombie or multiple. However over the icon to learn its meaning. The State column displays a question mark (?) when there are more than one process associated with the class. Expand the class to see the state of the related processes.
- **Effective User:** The name of the user account that started the process.
- **Start Time:** The time, as set on the Controller machine, when the process started.
- **End Time:** The time, as set on the Controller machine, when the process ended.
- **CPU (%):** The percentage of CPU resources by all process in this class.
- **CPU Trend:** A chart that shows CPU usage over the selected time period.
- **Memory (%):** The percentage of memory resources by all process in this class.
- **Memory Trend:** A chart that shows memory usage over the selected time period.
- **PID:** Process ID.
- **PPID:** ID of the parent process.

**OS Specific Columns:** The following column information is specific to processes monitored on a Linux server.

- **pgid:** Process group ID.
- **Real Group:** The process real user ID.
- **Real User:** The process real user ID.
- **Effective Group:** The user ID the kernel uses to determine the process permissions when using shared resources such as message queues, shared memory, and semaphores.
• **Nice Level**: The priority used to indicate the amount of CPU to afford the process or the process priority, where -20 is the highest priority and 19 or 20 is the lowest priority.

For more details of the metrics shown here, see [Server Monitoring Metrics Reference](#).

For information about configuring process monitoring, see [Configure Server Monitoring](#).

**View Server Metrics From Application Context**

**Related pages:**
- Monitor Your Servers using Server Monitoring
- Flow Maps

With Server Monitoring enabled you can access server performance metrics in the context of your applications.

After logging into the AppDynamics Controller, and locating your application flow map, you can drill down to Server metrics for a selected tier or node. The application flow map view also displays a summary of the Health rule status for the application’s servers. You can see if any Servers have critical or warning conditions.

Click a tier, such as in this example, Inventory-Services, to see the server details.

From the list of transaction snapshots, you can drill down to server metrics to determine if there is anything suspicious contributing to slow transactions. Double-click a snapshot of interest, drill-down into the call to see the server tab and associated metrics.
On the snapshot Server tab, you can review CPU, memory, and network utilization and examine which processes are consuming server resources.
This topic describes the additional metrics collected by the Standalone Machine Agent for Server Monitoring. Server Monitoring is currently available for Linux and Windows. The metrics can be viewed from the Servers tab of the AppDynamics Controller and from the Metric Browser.

Server Monitoring metrics can only be viewed in the Server Monitoring version of the Metric Browser. You see this when you access the Metric Browser from the Servers tab in the top navigation bar of the Controller.

For most types of metrics in the browser, you can click any of the points in the graph to view more information about the metric observed at that point in time. The information shown includes the metric identifier, date and time of the observation, along with any of the following values relevant to the metric:

- **Obs** (observed value): the average of all data points seen for that interval. For a cluster or a time rollup, this represents the weighted average across nodes or over time.
- **Min**: the minimum data point value seen for that interval
- **Max**: the maximum data point value seen for that interval
- **Sum**: the sum of all data point values seen for that interval. For the Percentile Metric for the App Agent for Java, this is the result of the percentile value multiplied by the Count.
- **Count**: the number of data points generated for the metric in that interval. This is information can be useful because the collection interval for infrastructure metrics varies by environment.

This reference describes metrics that are available only with Server Monitoring. The metrics are reported for both Windows and Linux servers unless otherwise noted. The basic hardware metrics are described in Machine Agent Hardware Metrics.

Some metrics that appear in the Hardware Resources branch of the Metric Browser are collected by the Database Agent and are used in the Database Monitoring UI, while others may be custom metrics added by extensions run by your Machine Agent.

### CPU Metrics

Note: These CPU metrics are not available for Windows Servers

- **%Nice**: % CPU time spent on low priority processes.
- **%Irq**: % CPU time spent servicing/handling hardware interrupts.
- **%Softirq**: % CPU time spent servicing/handling software interrupts

**System**: The percentage of time the CPU was busy processing kernel code.
**User**: The percentage of time the CPU was busy processing non-kernel code.

**IOWait**: The percentage of time the CPU spent waiting for an I/O request.

---

**Disks Metrics**

The disk metrics are reported for each disk partition and for all disks.

**Avg IO Utilization (%)**: The average time spent processing read/write requests as a percentage of the total reported time window. Databases often report high disk I/O utilization due to frequent read/write requests. For example, if the agent detects read/write processing in 55 out of 60 seconds, the Avg IO Utilization for that minute is 92%. This metric does not measure the amount of available disk space or read/write request sizes.

**Avg read time (ms)**: The average time required to service a read request.

**Avg write time (ms)**: The average time required to service a write request.

---

**For some versions of Windows Server 2008 and Windows Vista, the Hardware Resources|Disks|<partition>|Avg Read Time (ms) and Hardware Resources|Disks|<partition>|Avg Write Time (ms) metrics will be reported as 0. This is due to a known bug with Windows, and can be resolved by downloading the hotfix: [https://support.microsoft.com/en-us/kb/961435](https://support.microsoft.com/en-us/kb/961435)**

---

**Load Metrics**

The load metrics are reported for each machine.

The CPU % (reported as part of the basic Machine Agent metrics) is the percentage of the CPU consumed by processes that are currently running. Load takes into account processes that are waiting to run.

**Last 1 minute**: CPU Load, presented as an average.

**Last 5 minutes**: CPU Load, presented as an average over the last 5 minutes.

**Last 15 minutes**: CPU Load, presented as an average over the last 15 minutes.

---

**Machine Metrics**

**Availability**: The percentage of time the Machine Agent was reporting to the Controller. In the Server Monitoring UI, this provides a percentage with 6 digits of precision (i.e. 100.0000), in other words measuring up to 6 9's of availability. The Standalone Machine Agent sends a heartbeat to the AppDynamics Controller once per minute to indicate the agent is reporting.

Reported for each machine.

---

**Memory Metrics**

Reported for each hierarchical group.

**Swap Free (MB)**: The total amount of free swap space.

**Swap Total (MB)**: The total amount of allocated swap space.

**Swap Used (MB)**: The amount of swap space used.

**Swap Used %**: The percentage of used swap space.

**Swap Free %**: Percentage of free swap space.

---

**Network Metrics**
**Avg Utilization (%)**: The average network utilization as a percentage of the maximum possible throughput. This metric is not reported if the agent cannot determine the throughput (not supported for some devices and Linux versions). The percentage is rounded to the nearest integer. Therefore, very low utilizations might be reported as 0%. Reported for:

- Servers
- `<group>`
- `<network device>`

**Incoming Errors/min**: the rate of incoming packet errors the network incurs every minute. Not available for Linux Servers.

**Outgoing Errors/min**: the rate of outgoing packet errors the network incurs every minute. Not available for Linux Servers.

**Process Metrics**

Process metrics are aggregated and reported per process. The Servers Process window displays the Top 10 Processes Consuming CPU and Top 10 Processes Consuming Memory. To modify the default process monitoring, see Configure Server Monitoring.

- **Count**: The total number of processes in this class.
- **CPU Used (%)**: Percentage of the CPU bandwidth used by all processes in a process class. A process using 100% CPU is executing on all processors on the system.
- **Major Faults/sec**: Number of major page faults caused by all processes in this class.
- **Minor Faults/sec**: Number of minor page faults caused by all processes in this class.
- **Memory Used (%)**: Percentage of memory consumed by the top 10 consuming processes or the percentage of memory used by all processes in this class.
- **Memory Used (KB)**: Amount of memory used by all processes in this class.
- **Memory Virtual (KB)**: Current size of the virtual address space that the process is using. Use of virtual address space does not necessarily imply a corresponding use of either disk or main memory pages. Virtual space is finite and, by using too much, the process can limit its ability to load libraries.
- **Threads Count**: The number of kernel threads used by all processes in this class.

**Service Availability**

Service Availability metrics are reported for each service that is configured.

- **Response Time (ms)**: The elapsed time between sending a request and receiving a response from the monitored service.
- **Response Size (bytes)**: The size of the response received from the monitored service.
- **Success Rate (%)**: The percentage of successful requests over all requests made to the service.

**Volumes Metrics**

The volume metrics are reported across all listed volumes and for each volume at a specified mount point, such as `/boot`.

- **Total (MB)**: The amount of storage space available (used and free) across all listed volumes or at the specified mount point. On Linux, the space reserved for root is not counted in the available space.
- **Free (MB)**: The amount of unused or free space across all listed volumes or on the selected volume at the specified mount point. On Linux, the space reserved for root is not counted in the available space.
- **Used (MB)**: The amount of storage space in use across all listed volumes or on the selected volume at the specified mount point.
- **Used (%)**: The percentage of storage space in use across all listed volumes or on the selected volume at the specified mount point.
Service Availability Monitoring

The Service Availability feature of Server Monitoring enables you to monitor internal or external HTTP and HTTPS services. You configure service monitoring from the controller UI.

Once you have configured a service for monitoring, the Machine Agent checks the service according to your configuration, evaluating the response against your specified validation rules. When a validation rule is violated for a response, the response is categorized as a failed response, otherwise it is a successful response. The state of the service is evaluated based on the values you specify for the results window, success threshold and failure threshold. The results window is the number of check results that are monitored as a rolling window to evaluate the success and failure threshold. This window must be full before it is used to evaluate the state of the monitored service. Once the results window is full, the window is evaluated every time a new response is received.

For example with following values:

- Success threshold = 3
- Failure threshold = 1
- Results window size = 5

The agent waits until five check results are received and from these five results, if there is one failure then the target state is CRITICAL. If there are at least three successful results, then the target state is NORMAL.

License

Service Availability Monitoring requires a separate license and a Server Monitoring license. Server Monitoring is currently available for Linux and Windows. For more details, see License Information.

Viewing the Monitored Services

The monitored services can be viewed in the Monitored Services list. The state is determined as described above by evaluating the Response Validators during the results window. Response Validators are rules you configure that are used to evaluate against the responses received from the service.

The possible states are:

- NORMAL - The number of successful responses in the result window is greater than or equal to the configured success threshold.
- CRITICAL - The number of failed responses in the result window is greater than or equal to the configured failed threshold.
- UNKNOWN - If the machine agent does not provide any data.

Monitored Services List

From the top navigation bar, click Servers > Service Availability to see a list of monitored services. Use the action toolbar to manage your monitored services by adding new ones, viewing details and so on.
Monitored Service Details

Select a monitored service and double-click to see the associated details.

**Yahoo Error**

<table>
<thead>
<tr>
<th>Response Size</th>
<th>Response Time</th>
<th>Success Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>881 bytes</td>
<td>0.111 ms</td>
<td>0 %</td>
</tr>
</tbody>
</table>

**Events**

- 12/22/15 11:52:03 AM: Status Code 200, Response Time 105 ms, Response Size 881 bytes, Summary: Failed to validate: Response status code is not between 400 and 400 inclusive.
- 12/22/15 11:57:03 AM: Status Code 200, Response Time 103 ms, Response Size 881 bytes, Summary: Failed to validate: Response status code is not between 400 and 400 inclusive.
- 12/22/15 11:47:03 AM: Status Code 200, Response Time 107 ms, Response Size 881 bytes, Summary: Failed to validate: Response status code is not between 400 and 400 inclusive.
- 12/22/15 11:46:34 AM: Status Code 200, Response Time 144 ms, Response Size 881 bytes, Summary: Failed to validate: Response status code is not between 400 and 400 inclusive.

**Configure Service Availability**

By default, configuration changes take effect within at least two minutes of the change, so you do not need to restart the Machine Agent after changing the configuration. If the protocol, target, or method of a service are changed once a service is created, the state of the service is set to UNKOWN, and is re-evaluated with the new configuration.

To access the configuration window do the following:
1. From the Controller top navigation bar, select **Servers > Service Availability**.
   You see a list of monitored services if any are configured.

2. To add a service, click **Add**.
   You see the **Add Service Monitoring Configuration** window with three tabs.

3. On the Main tab, specify the required fields and save the configuration. Each field has an associated tool tip that provides help. For details, see **Main Configuration Window**.

4. On the Request Configuration tab, specify your request headers. For details, see **Request Configuration**.

5. On the Response Validators tab, configure your validation rules. For details, see **Response Validators**.

### Main Configuration Window

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Default</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of your choice for this target configuration. This name appears in the Service Availability list.</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Target</td>
<td>The resource to be monitored, for example, <code>http://myThirdPartyService.com/data</code>. Specify which HTTP method to use to send the request (GET, POST, HEAD)</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Server</td>
<td>Name of the machine agent performing the monitoring. Only servers that are enable for Server Monitoring show in the drop-down list.</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Socket Timeout</td>
<td>How long to wait, in milliseconds, after a successful connection for a complete HTTP response.</td>
<td>30000</td>
<td>Yes</td>
</tr>
<tr>
<td>Success threshold</td>
<td>Number of required successes within the results window for a NORMAL state.</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>Check Interval</td>
<td>Interval time in between checks in seconds.</td>
<td>10</td>
<td>Yes</td>
</tr>
<tr>
<td>Results Window Size</td>
<td>Number of most recent results to use in determining the state of the service.</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>Connect Timeout</td>
<td>How long to wait (in ms) for the service to respond to a connection request.</td>
<td>30000</td>
<td>Yes</td>
</tr>
<tr>
<td>Failure threshold</td>
<td>Number of required failures within the result window for a CRITICAL state.</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>Max response size</td>
<td>Maximum response size to collect in bytes.</td>
<td>5000</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Follow redirects | Follow redirect to determine state of service. | Yes | No

**Request Configuration**

You can define a list of customized headers to send with requests. For example, the list can mimic desktop or mobile browsers. You can also define a request body for POST requests. An "Authorization" header can be specified if credentials are required.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Field</th>
<th>Value Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>header name and value pair</td>
<td>String</td>
<td>empty</td>
<td>Customized header to send with requests.</td>
</tr>
<tr>
<td></td>
<td>body</td>
<td>String</td>
<td>empty</td>
<td>Any data to send with the request. Usually used for POST.</td>
</tr>
</tbody>
</table>

**Response Validators**

You can provide a list of rules for the machine agent to use to validate whether the monitored service is healthy. If any rule is violated, the response is considered failed.

For each rule you can specify the following:

- HTTP property
- operator
- a value

The list of operators varies depending on the property selected in the first drop-down.

**Monitoring the Service Health**

An event is sent to the controller for every state change and periodically (by default every 5 minutes). The update interval can be changed by setting the System property:
appdynamics.machine.agent.sam.event.updateIntervalMillis. For more detail, see "Service Availability Update Interval" in the Configuration Property Reference.

The events are visible from the Monitored Service details window and from the Server Monitoring Events list.

You can see details by double-clicking the event.

The event properties are as follows:

- Category: CUSTOM
- Event Type: Service Availability
- Property: Severity values can be ERROR (failure) or INFO (passing).

Configuring Alerts

Use these steps to create alerts for your monitored services:

1. From the Servers window, click Alert & Respond.
2. Select Policies.
3. Select Create Policy Manually.
   If policies already exist, you may not see this option. Just click + to add a policy manually.
4. Under Custom Events, click + to Add Custom Event.
5. In the Custom Event Filter, add the following:
   a. Set the Event Type to “Service Availability”.
   b. To filter on failure, add the severity = ERROR property.
   c. (Optional) To apply the customer filter to a specific service, add the property: targetID = <target_id>, which is the ID of the monitored service to which the policy applies. If you do not specify this property, the customer filter is applied to all monitored services.
      To view the target ID, go to Servers > Service Availability > View Options and enable Show Config id. Then check the Config ids field for the relevant service.

6. In the Actions section of the Policy window, add the action you want to execute if the policy violates and save the policy.

Server Monitoring Events

On this page:
- Custom Events
- Agent Internal Diagnostic Events

Related pages:
- Filter and Analyze Events
- Service Availability Monitoring

You can see Machine Agent events on the Server Events list and on the Application Events list for applications associated with the machine agent.
Access the Servers Events list from the top navigation bar, click Servers > Events.
To access the Application Events list, see Monitor Events.

Machine Agent events include the following:

- Custom Events
  - APPDYNAMICS_INTERNAL_DIAGNOSTICS type: EventMessage.MACHINE_AGENT_RESET_SUCCEEDED

**Custom Events**

Custom events include events generated by the extensions and Service Availability Monitoring events. Service Availability Monitoring events are custom events that are sent to the controller periodically and for every state change in the service's health.

Events triggered by extensions are custom events and also display on the Servers > Events list.

If you want to view custom events in the application Events list, you can add them to the event list filters.
ACME Book Store Application

Agent Internal Diagnostic Events

AGENT_DIAGNOSTICS

Description: Diagnostic information concerning agent activity, such as resetting a machine agent. The reset functionality works only for pre 4.2 Machine Agents.

Category: AppDynamics Internal Diagnostics

Name: EventMessage.MACHINE_AGENT_RESET_SUCCEEDED
Configure Server Monitoring

On this page:
- Editing YML Files: Important Notes
- Configurable Server Visibility Settings
- Process Limits

You can modify the default configuration for Server Monitoring. This page describes the configurable settings and their default values.

Configuration file to edit: `<machine-agent-home>/extensions/ServerMonitoring/conf/ServerMonitoring.yml`

Editing YML Files: Important Notes

Note the following:
- If you make any changes to ServerMonitoring.yml, or to any other YML file, make sure that the modified file meets standard YML syntax rules. Some important YML good practices include:
  - Do not include any tab characters. Use whitespace characters only to indent fields.
  - Use the exact same number of whitespace characters to indent entries that are on the same level.
  - Use a plain-text editor, not a WYSIWIG editor, to edit the file. Use a monowidth/monospaced font to view the contents.
  - Always save using UTF-8 encoding.
  - Test and verify the edited file using an online YML syntax tester.
- The safest way to edit a setting in this file is to
  - Copy the line you want to edit and paste the copy into a new line. Make sure that you select, copy, and paste the entire line.
  - Comment out the original line and edit the copy as desired.
- If you want to add or edit a regular expression in this file, it is good practice to test and verify the regular expression using an online regex tester.
- Before you make any changes to this file, read the option descriptions below and the comments in the file carefully for the setting you want to change.

Configurable Server Visibility Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>volumeMonitorConfig:</td>
<td></td>
</tr>
<tr>
<td>maxNumberVolumes</td>
<td><em>(new in 4.2.4)</em> Do not report more than N volumes, where N = maxNumberVolumes. Default = 5</td>
</tr>
<tr>
<td></td>
<td>Changing this setting can affect the resource consumption of your deployment. Before you increase this setting, verify that your application environment and Controller can handle the increased resource requirements.</td>
</tr>
<tr>
<td>volumeMonitorConfig:</td>
<td><em>(new in 4.2.4)</em> Volumes with names that match this regular expression are always reported, up to the maximum specified by maxNumberVolumes. Default = &quot;&quot;</td>
</tr>
<tr>
<td>whitelistSelectorRegex</td>
<td></td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>volumeMonitorConfig</strong>:</td>
<td>(new in 4.2.4) Volumes with names that match this regular expression are excluded. This setting is useful for filtering out irrelevant metrics. If a volume name matches both the blacklist and whitelist regexes, metrics for that network are reported (the whitelist takes priority) up to the maximum specified by maxNumberVolumes. The default ServerMonitoring.yml file does not include a blacklistSelectorRegex field for volumes. If you want to add one, use the same indentation and formatting as the volumeMonitorConfig:whitelistSelectorRegex field.</td>
</tr>
<tr>
<td>blacklistSelectorRegex</td>
<td></td>
</tr>
<tr>
<td>maxNumberVolumes</td>
<td>Do not report more than N networks, where N = maxNumberNetworks. Default = 5</td>
</tr>
<tr>
<td><strong>networkMonitorConfig</strong>:</td>
<td>(new in 4.2.4) Networks with names that match this regular expression are always reported, up to the maximum specified by maxNumberNetworks. To report metrics for one or more virtual networks, specify a regex that matches the virtual network names. Default = &quot;&quot;</td>
</tr>
<tr>
<td>maxNumberNetworks</td>
<td></td>
</tr>
<tr>
<td>whitelistSelectorRegex</td>
<td>(new in 4.2.4) Networks with names that match this regular expression are excluded. This setting is useful for filtering out irrelevant metrics. If a network name matches both the blacklist and whitelist regexes, metrics for that network are reported (the whitelist takes priority) up to the maximum specified by maxNumberNetworks. The default regex excludes virtual networks. To monitor a set of one or more virtual networks, edit whitelistSelectorRegex to include the networks of interest. To monitor all virtual networks, change the blacklistSelectorRegex to an empty string. Default = &quot;^veth.*</td>
</tr>
<tr>
<td>blacklistSelectorRegex</td>
<td></td>
</tr>
<tr>
<td><strong>defaultProcessClassSelector</strong></td>
<td>(new in 4.2.4) The default &quot;class selector&quot; based on a class_selector_regex. If this regex is defined, and a match is found in the process command line, the class name is the first group occurrence of that regex in the command line. Default = &quot;&quot;</td>
</tr>
</tbody>
</table>
### processClassSelectorRegexList

(new in 4.2.4) A list of `class_name:regex` mappings. If the command line for a process matches `regex`, the metrics for that process are assigned to `class_name`. This setting is useful when you want to ensure that high-priority processes get reported, even if the number of defined classes is higher than the maxNumberMonitoredClasses setting.

Here is an example:

```yaml
processMonitorConfig:
  processClassSelectorRegexList:
    machineAgentTasks: '.*java.*machineagent.*'
    controllerTasks: '.*java.*controller.*'
    nextOne: '.*svchost.*'
```

The Machine Agent assigns a process to a class as follows:

1. Assign to the first match in `processClassSelectorRegexList`.
2. If there is no match for the `processClassSelectorRegexList` (step 1), apply the `defaultProcessClassSelector` regex to the command line.
3. If there is no match for the `defaultProcessClassSelector` regex (step 2), use the process name (truncated if the name exceeds the `maxClassIdLength`).

The following steps outline the recommended workflow for updating this list:

1. The default ServerMonitoring.yml file includes an example that is commented out. If you are updating the default list for the first time, it is good practice to
   a. Create a copy of the example.
   b. Uncomment the copy (remove the <!-- and --> comment tags) and edit it as needed.
2. Choose the process(es) that you want to monitor on the host machine.
3. Create a regex to match the process name(s) of interest. It is good practice to test the regex using an online regex validator.
4. Add the regex to the list. You should order the mappings by priority, highest to lowest. If a command line matches multiple regexes, the first match is used.
5. Save the ServerMonitoringConfiguration.yml file.
6. Wait 15 minutes or longer for the updated list to take effect, then verify that the matching process(es) appear in the Controller UI.

### samplingInterval

Indicates how often to gather metric data. Units in milliseconds.

Default = 60000 (60 seconds)

### maxClassIdLength

Specifies the maximum process class name length. Any process class name that is longer than the specified maximum is truncated. The global maximum of the process class name is 100. If this variable is set to be greater than 100, then the process name is truncated at 100.

Default = 50

### processSelectorRegex

Contains a regular expression that specifies which processes should be monitored by the machine agent. The regular expression is compared against the full command line that was used to start the process. The default expression filters out

- Linux processes for which the arguments could not be found. These are usually kernel processes.
- Windows processes with blank command lines.

Default = "[^\[\]]*$"

### minLiveTimeMillisBeforeMonitoring

Specifies the minimum amount of time a process must be alive before it is monitored by the machine agent. This is useful for preventing the machine agent from being overloaded by monitoring short-lived processes. Units in milliseconds.

Default = 60000 (60 seconds)
maxNumberMonitoredClasses | Specifies the maximum number of process classes that the machine agent monitors. If the number of classes in processClassSelectorRegexList is higher than the maxNumberMonitoredClasses, the agent selects the classes to monitor based on the order of processClassSelectorRegexList (highest-priority processes first). Set this field to 0 to turn off all process monitoring.

For example, assume the following scenario: maxNumberMonitoredClasses = 2 and processClassSelectorRegexList has three classes (A, B, and C, in that order). The machine has 25 processes running:

- Class A has 12 processes
- Class B has 5 processes
- Class C has 8 processes

In this scenario the agent sends metrics and metadata for the first two classes, A and B (17 processes in all), to the Controller. Class C is not in the first two listed classes, so these processes are not monitored.

Default = 20

basicEnabled | Indicates whether the machine agent should report the basic metrics through the SIM extension. Set this to false to use Sigar to report basic metrics. This setting only affects monitoring on Linux.

Default = true on Linux, false on Windows

defaultDiskSectorSize | Specifies the default sector size (in bytes) for each disk if the machine agent cannot determine the sector size. This value is used to calculate the number of bytes read/written for the disk.

Default = 512

Process Limits

Other configurable settings are available in the Controller Admin UI, for example:

- **sim.processes.query.maxResultLimit**: The total number of processes displayed in the UI for a single query is 5000 processes per call.
- **sim.machines.count.maxPerAccount**: The total number of processes tracked per account. The default value is 10000 processes per account.

For more information, see Controller Settings for Server Monitoring.

Configure Machine Health Rules, Policies, and Alerts

On this page:
- Health Rules
- Policies
- Actions
- Email Digests

Related pages:
- Health Rules
- Alert and Respond
- Policies
- Health Rules
- Actions
You can configure Server Monitoring to alert you when certain conditions are met or exceeded based on monitored server hardware metrics. The Getting Started Wizard can help you through the process if you’re new to using AppDynamics Pro. Alternatively you can create alerting policies manually if you are already familiar with the process.

You configure Health Rules, Actions, Policies, and Email Digests for monitoring servers almost exactly the same as you would configure these for monitored applications, with some minor changes.

**Health Rules**

Default health rules based on several usage metrics are enabled for Server Monitoring. You can modify these and set up your own health rules. The default health rules are shown in the following screen shot:

![Health Rules Table](image)

On the Affects tab of the Health Rule wizard, you can choose to apply the Server Health Rule to:

**Subgroups**
- all or selected subgroups.

**Machines**
- all machines in the active account
- machines within selected subgroups
- selected machines
- machines whose names match certain criteria

The health rule is violated when specified critical or warning conditions are met.
Hierarchies, Groups, and Subgroups

Health rules can be applied to subgroups or machines within selected subgroups. Subgroups available on the Affects tab of the Health Rules wizard are Machines that are grouped into hierarchies depending on the Machine Hierarchy Property configuration for the Standalone Machine Agent. Also, see Machine Agent Hierarchy. The subgroups are the leaf group a machine is in. So for example, if there are three machines where:

- A is in group Data Center 1|Rack 1
- B is in group Data Center 1|Rack 2
- C is in group Data Center 1|Rack 1

A and C are in the same subgroup, B is in a different one.

Policies

Server Policy Actions are triggered when any or select Health Rule Violation Events occur. Unlike Application Policies, Server Policies cannot be based on custom events.

Actions

Actions are performed when a health rule is violated and a policy is triggered. Actions can be email or SMS message notifications, HTTP requests or custom actions that have been uploaded to the Controller. There are also email and HTTP request templates for you to use for alert and respond actions.

Email Digests

Email Digests let you send notification of chosen health rule violation events to specific email addresses at a specified frequency, such as every 2 hours.

Controller Settings for Server Monitoring

<table>
<thead>
<tr>
<th>Related pages:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Access the Administration Console</td>
</tr>
<tr>
<td>- Database Size and Data Retention</td>
</tr>
</tbody>
</table>

This topic describes Controller Admin settings that are specific to Server Monitoring. You need the root user password to change these settings. For details on how to change the Controller Admin settings, see Controller Settings for Standalone Machine Agents.

<table>
<thead>
<tr>
<th>Server Monitoring Property</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| sim.metrics.store.customMetrics.simNodeEnabled | (New in 4.2.4) Enables or disables the persistence of custom metric values to machines monitored by Server Visibility. This setting works as follows:  
  • true—The collector stores and displays custom metric values for applications and machines. Custom metrics are displayed in Metric Browsers for both Applications and Server Visibility.  
  • false—The collector stores and displays custom metric values for applications only. Custom metrics are displayed in the Metric Browser for Applications only (not for Server Visibility).  
You might want to disable this setting if you are collecting a lot of custom metrics. This will prevent potential spikes in the number and rate of metric values stored for individual machines. | true  |
| sim.machines.agent.process.maxClasses       | (New in 4.2.4) Global maximum for the number of process classes collected per machine.  
  • If maxClasses (global setting in Controller) is lower than maxNumberMonitoredClasses (local setting in serverMonitoringConfig.yml), the local setting is overridden and the global MaxClasses is the effective maximum for that machine.  
  • if maxClasses (global) is higher than maxNumberMonitoredClasses (local), the local setting is the effective maximum for that machine.  
For information about local settings, see Configure Server Monitoring. | 20    |
| sim.processes.count.maxPerMachine           | (New in 4.2.4) Global maximum for the number of processes collected per machine.  
Increasing this setting can affect the resource consumption of your deployment. Before you increase this setting, verify that your application environment and Controller can handle the increased resource requirements. | 1000  |
<p>| sim.exceptions.stacktrace.enabled           | When this is enabled, controller stacktraces are sent in the error response when a client request encounters an error. Setting this to true is a security risk, so should only be set to true if you understand the consequences and if directed by a support engineer. | false |</p>
<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim.machines.registrations.maxPerSecondPerAccount</td>
<td>Maximum number of registrations allowed per second per account. This setting controls the rate at which concurrent registrations are processed and prevents one account on a multi-tenant controller from monopolizing resources needed to register the agents. The default value assumes that the machine agents are started evenly across a minute. If you are running 2000 or more machines, you may need to increase this setting. A suggested formula is ((number of 4.2+ machine agents / 60) * 8). If you use a deployment script that starts all your agents within a few seconds of each other, you may need to further adjust the max registrations per second.</td>
<td>60</td>
</tr>
<tr>
<td>sim.machines.count.maxPerAccount</td>
<td>Maximum number of machines allowed per account. Any additional machines will not appear. If you have more than 2000 machines reporting to one controller and need to increase this number, be aware that you might need to increase the sim.machines.registrations.maxPerSecondPerAccount setting as well.</td>
<td>2000/Account</td>
</tr>
<tr>
<td>sim.machines.offline.toStaleTimeoutMillis</td>
<td>How much time, in milliseconds, to wait before considering an offline machine to be stale and marked for deletion. If this value is too high, it prevents fresh data from coming in. If the value is too short means less history.</td>
<td>2592000000 ms (30 days)</td>
</tr>
<tr>
<td>sim.machines.stale.purgeIntervalMillis</td>
<td>An interval in milliseconds that determines when stale machines are deleted from the controller database. If this value is too short, it might overload the server. If the value is too high, then stale machines are deleted more slowly.</td>
<td>21600000 ms (6 hours)</td>
</tr>
<tr>
<td>sim.machines.lastSeenTimestamp.updateIntervalMillis</td>
<td>This setting controls the granularity of the &quot;last-seen&quot; timestamp that is used to check if a machine is offline or not. This dictates the minimum value for sim.machines.offline.toStaleTimeoutMillis. Smaller values mean more database writes. Unit is milliseconds.</td>
<td>3600000 ms (1 hour)</td>
</tr>
<tr>
<td>sim.machines.simAllowed</td>
<td>This setting allows you to enable or disable Server Monitoring from the controller.</td>
<td>true</td>
</tr>
<tr>
<td>sim.processes.query.maxResultLimit</td>
<td>The maximum number of processes that can be viewed in the UI in a single query. The default is 5000 processes. For example, viewing the Processes Detail page, if the time range is set to 2 weeks and there are total of 10,000 processes that were reported over the last 2 weeks, the UI can only display 5000 processes.</td>
<td>5000</td>
</tr>
<tr>
<td>sim.processes.registrations.maxPerSecondPerAccount</td>
<td>Maximum number of process requests handled per second per account.</td>
<td>60/sec</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td><code>sim.processes.terminated.toStaleTimeoutMillis</code></td>
<td>Number of milliseconds before a terminated process is considered stale and can be deleted to make space for new data.</td>
<td>604800000 ms (7 days)</td>
</tr>
<tr>
<td><code>sim.processes.count.maxPerAccount</code></td>
<td>Maximum number of processes stored per account.</td>
<td>300000</td>
</tr>
<tr>
<td><code>sim.processes.creation.maxConcurrent</code></td>
<td>The maximum number of processes that can be registered simultaneously by the controller. The default is 5000 processes. The machine agent will retry to register the processes if the first attempt fails. Until the registration request is accepted and processed by the Controller, processes within that request will be missing from the Process Details page. The limit works as follows: Case 1: Machine Agent sends a request with 6000 processes to register, and the limit is set to 5000, then this request is rejected until the limit is increased. Case 2: Two Machine Agents (MA1 and MA2) each try to register 3000 processes and the limit is set to 5000. Both MA1 and MA2 requests are received by the Controller and only one of the requests will be processed (3000 + 3000 &gt; 5000). So, Controller can only process one request, and rejects the other one. Case 3: Two Machine Agents (MA1 and MA2) each try to register 500 processes and the limit is set to 5000. Both requests arrived at the Controller and since 500 + 500 &lt; 5000, then both requests are processed and the processes are registered.</td>
<td>5000</td>
</tr>
<tr>
<td><code>sim.processes.stale.purgeIntervalMillis</code></td>
<td>Number of milliseconds between consecutive deletes of stale processes for an account.</td>
<td>21600000 ms (6 hrs)</td>
</tr>
<tr>
<td><code>sim.machines.hostidMappingAllowed</code></td>
<td>You must enable this mode if you want to collect and view Machine or Server metrics on a server with Machine and .NET Agents installed. For additional requirements and important notes, see .NET Compatibility Mode.</td>
<td>false</td>
</tr>
</tbody>
</table>

**Machine Agent Hierarchy**

**Related pages:**
- Standalone Machine Agent Configuration Property Reference
- Configure Machine Health Rules, Policies, and Alerts

To group servers together so that health rules can be applied to the specific server groups, use the Machine Agent Hierarchy property. This property enables you to group servers into arbitrary hierarchies by specifying a hierarchical path to the server. Servers can be grouped based on departments, geographic locations such as data centers, or other organizational units. You can then create health rules that apply to these departments.

The server hierarchy displays in the Metric Browser, on the Servers list, and on Server Monitoring dashboard.

You need a Server Monitoring license to use this feature.

The property can be specified using controller-info.xml, a system property, or an environment variable. For details on the property, see Machine Hierarchy on this page: Standalone Machine Agent Configuration Property Reference.

To achieve something like this where you group servers into geo locations, perhaps representing different data centers:
Use the machine-path element in controller-info.xml configuration file, for example:

```xml
<machine-path>AWS|EC2|USEast|SIM-01</machine-path>
```

Metric Browser view for this example:

When creating health rules, on the Affects panel you can select one or more subgroups.