Infrastructure Visibility

Overview of Infrastructure Visibility

Hardware Resources Metrics

Standalone Machine Agent

Standalone Machine Agent Requirements and Supported Environments

Machine Agent Metric Collection

Monitoring Windows -- Guidelines and Important Notes

Configure Metrics for Virtual Disks and External Network Traffic - Java Hardware Monitor Extension Only

Viewing Hardware Metrics

Install the Standalone Machine Agent

Standalone Machine Agent Installation Scenarios

Plan the Machine Agent Configuration

Install Using the Non-JRE Zip File

Linux Install Using the RPM Package

Linux Install Using ZIP with Bundled JRE

Solaris Install Using ZIP with Bundled JRE

Windows Install Using ZIP with Bundled JRE

Deploy Multiple Machine Agents From a Common Directory

Verify the Standalone Machine Agent Installation

Resolve Standalone Machine Agent Installation Problems

Start and Stop the Standalone Machine Agent

Permissions Required to Run the Machine Agent

Configure the Standalone Machine Agent

Where to Specify Machine Agent Configuration

Standalone Machine Agent Configuration Properties

Controller Settings for Standalone Machine Agents

Enable SSL for Standalone Machine Agent

.NET Compatibility Mode

Extensions and Custom Metrics

Build a Monitoring Extension Using Scripts

Build a Monitoring Extension Using Java

Standalone Machine Agent HTTP Listener

Administer the Standalone Machine Agent

Manage Machine Agents

Upgrade the Standalone Machine Agent

Uninstall the Standalone Machine Agent

FAQs and Troubleshooting

Server Visibility

Server Visibility Requirements and Supported Environments

Enable Server Visibility

Monitor Your Servers using Server Visibility

Discover Normal Server Activity

Servers List

Server Dashboard

Server Process Metrics

Server Volumes Metrics

Server Network Metrics

Navigating Between Server and Application Contexts

Configuring Docker Visibility

Dynamic Monitoring Mode and Server Visibility

Tier Metric Correlator

Service Availability Monitoring

Server Visibility Events

Machine Agent Settings for Server Visibility

Configure Health Rules to Monitor Servers

Controller Settings for Server Visibility

Machine Agent Hierarchy

Server Tagging

Tagged Metrics

Network Visibility

Network Visibility Overview

Network Visibility Requirements and Supported Environments

Set Up Network Visibility on Linux

Install the Network Agent on Linux

Set Up the Network and App Agents on Linux

Administer the Network Agent on Linux

ZIP Network Agent Operations

RPM Network Agent Operations

DEB Network Agent Operations

Managing Network Agents in the Controller

Network Visibility with Kubernetes

Set up Network Visibility on Windows
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the Network Agent on Windows</td>
<td>202</td>
</tr>
<tr>
<td>Install the App Agent on Windows</td>
<td>204</td>
</tr>
<tr>
<td>Administer the Network Agent on Windows</td>
<td>206</td>
</tr>
<tr>
<td>Network Visibility Metrics</td>
<td>208</td>
</tr>
<tr>
<td>KPI Metrics in Network Dashboard and Application Flow Map</td>
<td>209</td>
</tr>
<tr>
<td>KPI Metrics in Right-Click Dashboards</td>
<td>211</td>
</tr>
<tr>
<td>Node Metrics in Network Dashboard and Metric Browser</td>
<td>215</td>
</tr>
<tr>
<td>TCP Connection Metrics in Metric Browser</td>
<td>217</td>
</tr>
<tr>
<td>FAQs for Network Visibility</td>
<td>223</td>
</tr>
<tr>
<td>Network Visibility Concepts</td>
<td>226</td>
</tr>
<tr>
<td>Identifying Performance Bottlenecks</td>
<td>227</td>
</tr>
<tr>
<td>Load Balancers and TCP Endpoints</td>
<td>228</td>
</tr>
<tr>
<td>Flows, Links, and Connections</td>
<td>230</td>
</tr>
<tr>
<td>Dynamic Monitoring Mode and Network Visibility</td>
<td>232</td>
</tr>
<tr>
<td>Workflows and Example Use Cases</td>
<td>234</td>
</tr>
<tr>
<td>Network Congestion - Example Use Case</td>
<td>241</td>
</tr>
<tr>
<td>Packet Loss - Example Use Case</td>
<td>247</td>
</tr>
<tr>
<td>TCP Port Exhaustion - Example Use Case</td>
<td>252</td>
</tr>
<tr>
<td>TCP Window Bottlenecks - Example Use Case</td>
<td>255</td>
</tr>
<tr>
<td>Traffic Throttling - Example Use Case</td>
<td>261</td>
</tr>
<tr>
<td>User Interface</td>
<td>264</td>
</tr>
<tr>
<td>Network Dashboard</td>
<td>265</td>
</tr>
<tr>
<td>Connection Explorer</td>
<td>268</td>
</tr>
<tr>
<td>Custom Dashboards for Network Visibility</td>
<td>270</td>
</tr>
<tr>
<td>Network Visibility Events</td>
<td>271</td>
</tr>
<tr>
<td>Advanced Operations</td>
<td>273</td>
</tr>
<tr>
<td>Packet Captures</td>
<td>274</td>
</tr>
<tr>
<td>Resolving Unmapped Connections</td>
<td>277</td>
</tr>
<tr>
<td>Troubleshooting Network Visibility Problems</td>
<td>278</td>
</tr>
<tr>
<td>Monitoring Containers</td>
<td>279</td>
</tr>
<tr>
<td>Monitoring Kubernetes</td>
<td>285</td>
</tr>
<tr>
<td>Monitoring Red Hat OpenShift</td>
<td>293</td>
</tr>
<tr>
<td>Monitoring Pivotal Cloud Foundry</td>
<td>301</td>
</tr>
<tr>
<td>Container Metrics</td>
<td>306</td>
</tr>
</tbody>
</table>
Infrastructure Visibility

AppDynamics Infrastructure Visibility gives you end-to-end visibility on the performance of the hardware running your applications. Infrastructure Visibility helps you identify and troubleshoot problems that can affect application performance such as server failures, JVM crashes, and network packet loss. Infrastructure Visibility provides metrics such as:

- CPU busy/idle times, disk and partition reads/writes, and network interface utilization (Machine Agents)
- Packet loss, round trip times, connection setup/teardown errors, TCP window size issues, and retransmission timeouts (Network Visibility, additional license required)
- Disk/CPU/memory utilization, process, and machine availability (Server Visibility, additional license required)

### Machine Agent
- Requirements and Supported Environments
- Install the Agent
- Configure the Agent
- Enable SSL for the Agent
- Administer the Agent
- Extensions and Custom Metrics
- Hardware Resources Metrics
- Machine Agent Metric Collection

### Server Visibility
- Requirements and Supported Environments
- Monitoring Servers
- Configuring Docker Visibility
- Configuring Health Rules to Monitor Servers
- Tier Metric Correlator
- Server Tagging
- Service Availability Monitoring

### Network Visibility
- Requirements and Supported Environments
- Set Up Network Visibility on Linux
- Administer the Network Agent on Linux
- Set up Network Visibility on Windows
- Administer the Network Agent on Windows
- Network Visibility Metrics
- FAQs and Troubleshooting
- Workflows and Use Cases
- Advanced Operations

### Container Visibility
- Monitoring Containers
- Container Metrics
- Monitoring Kubernetes
Overview of Infrastructure Visibility

Many times the root cause of application issues is most obvious by looking at application, network, server, and machine metrics that measure infrastructure utilization. For example, the following infrastructure issues can slow down your application:

- Too much time spent in garbage collection of temporary objects (application metric)
- Packet loss between two nodes that results in retransmissions and slow calls (network metric)
- Inefficient processes that result in high CPU utilization (server metric)
- Excessively high rates of reads/writes on a specific disk or partition (hardware metric)

**Infrastructure Visibility** enables you to isolate, identify, and troubleshoot these types of issues. Infrastructure Visibility is based on a machine agent that runs with an app server agent on the same machine. These two agents provide multi-layer monitoring as follows:

1. The app server agent collects metrics about applications and identifies applications, tiers, and nodes with slow transactions, stalled transactions, and other application-performance issues.
2. The Network Agent monitors the network packets sent and received on each node and identifies lost/retransmitted packets, TCP bottlenecks, high round trip times, and other network issues.
3. The machine agent collects metrics at two levels:
   a. Server Visibility metrics about local processes, services, and resource utilization.
   b. Basic machine metrics about disks, memory, CPU, and network interfaces.

This multi-layer monitoring enables you to find possible correlations between application issues and service, process, hardware, network, or other issues on the machine.
Network Visibility

Network Visibility monitors traffic flows, network packets, TCP connections, and TCP ports. Network Agents leverage the APM intelligence of App Server Agents to identify the TCP connections used by each application. Network Visibility includes the following functionality:

- Detailed metrics about dropped/retransmitted packets, TCP window sizes (Limited / Zero), connection setup/teardown issues, high round trip times, and other performance-impacting issues
- Network Dashboard that highlights network KPIs (Key Performance Indicators) for tiers, nodes, and network links
- Right-click dashboards for tiers, nodes, and network links that enable quick drill-downs from transaction outliers to network root causes
- Automatic mapping of TCP connections with application flows
- Automatic detection of intermediate load balancers that split TCP connections
- Diagnostic mode for collecting advanced diagnostic information for individual connections

Server Visibility

Server Visibility monitors local processes, services, and resource utilization. You can use these metrics to identify time windows when problematic application performance correlates with problematic server performance on one or more nodes.

Server Visibility is an add-on module to the Standalone Machine Agent. With Server Visibility enabled, the machine agent provides the following additional functionality:

- Extended hardware metrics such as machine availability, disk/CPU/virtual-memory utilization, and process page faults
- Monitor application nodes that run inside Docker containers and identify container issues that impact application performance
- The Tier Metric Correlator, which enables you to identify load and performance anomalies across all nodes in a tier
- Import and define server tags that make it easy to query, filter, and compare related servers using custom metadata
- Monitor internal or external HTTP and HTTPS services
- Support for grouping servers so you can apply health rules to specific server groups
• Support for defining alerts that trigger when certain conditions are met or exceeded based on monitored server hardware metrics

Basic Machine Metrics

The Standalone Machine Agent collects basic hardware metrics from the server’s OS. This agent provides the following functionality:

• Basic hardware metrics from the server’s OS such as CPU and memory utilization, throughput on network interfaces, and disk and network I/O
• Support for creating extensions to generate custom metrics
• Support for running remediation scripts to automate your runbook procedures. You can optionally configure the remediation action to require human approval before the script is started.
• JVM Crash Guard for monitoring JVM crashes and optionally running remediation scripts

Java and .NET Infrastructure Monitoring

Infrastructure Visibility uses different agents to monitor Java and .NET environments:

• The Java Agent collects metrics for business applications and JVMs. The Standalone Machine Agent collects Server Visibility and hardware/OS metrics.
• The .NET Agent collects metrics for business applications and instrumented CLRs. The .NET Agent includes a .NET Machine Agent that collects IIS and hardware/OS metrics (see Monitor Windows Hardware Resources). The Standalone Machine Agent collects Server Visibility metrics.

Infrastructure Visibility Strategies

You can use the following strategies to find infrastructure issues that affect application performance:

• Transaction snapshots for slow or stalled transactions – You can use snapshots to correlate infrastructure metrics for the specific node so that you can identify the root cause of slow or stalled transactions.
• Metric correlation –
  • One example workflow is to open the Node Dashboard for a mission-critical server with a machine agent installed and then to cross-compare data in the following tabs:
    • JVM (application performance)
    • JMX (server performance)
    • Server (hardware resource consumption)
  • The Network Dashboard includes right-click dashboards for tiers, nodes, and network links. These dashboards make it easy to find correlations between application issues and network root causes.
  • The Tier Metric Correlator enables you to identify load and performance anomalies in a tier composed of a cluster of nodes running on containers or servers.
• Health rules – You can configure health rules on metrics such as garbage collection time, connection pool contention, or CPU usage to catch issues early in the cycle before there is an impact on your business transactions.
• Infrastructure rules, policies, and alerts – You can
• Create health rules on metrics such as garbage collection time, connection pool contention, or CPU usage to catch issues early in the cycle before there is an impact on your business transactions.
• Define policies that trigger actions (such send an email, start diagnostics, or perform a thread dump) when infrastructure metrics report a critical level.
• You can configure alerts for JVM and CLR crashes respectively using JVM Crash Guard and the .NET Machine Agent.
• Configure the agent to run scripts in response to critical events (for example, restart an application or JVM in response to a crash).

With the right monitoring strategy in place, you can be alerted to problems and fix them before user transactions are affected.
Hardware Resources Metrics

On this page:
- CPU Metrics
- Memory Metrics
- Disk and Partition Metrics
- Volume Metrics
- Load Metrics
- Machine Metrics
- Network Metrics
- Process Metrics
- Service Availability Metrics
- Enable Percentile Metric Reporting
- Configure Disk Usage Metric Collection on Linux
- Configure Free/Used Memory Metric Collection on Linux
- Viewing Server Visibility Metrics

Related pages:
- Viewing Hardware Metrics
- Monitor Your Servers using Server Visibility
- Dynamic Monitoring Mode and Server Visibility
- Network Visibility Metrics

This topic describes the basic hardware metrics collected by the Machine Agent and the additional metrics collected by the Standalone Machine Agent for Server Visibility. Not all of the metrics that appear in the Hardware Resources branch of the Metric Browser tree are provided by the Machine Agent; some are collected by the Database Agent and are used in the Database Monitoring UI, while others might be custom metrics added by another extension run by your Standalone Machine Agent.

For most metrics in the Metric 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 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 information can be useful because the collection interval for infrastructure metrics varies by environment. For example:
  - For AIX, HP-UX, Mac OS X and Z/OS, CPU and memory metrics are gathered every two seconds and averaged over a period of one minute. Machine agent network and disk metrics are gathered at one-minute intervals.
  - For Windows, Linux, and Solaris, all metrics are collected on one-minute intervals and aggregated over one-minute intervals.

### CPU Metrics

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
<th>Basic or Server Visibility</th>
<th>Windows</th>
<th>Linux</th>
<th>Solaris</th>
<th>AIX</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Idle</td>
<td>Percentage of time the CPU was idle; the CPU had completed its tasks and has nothing to do.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Diagnostic</td>
</tr>
<tr>
<td>%Busy</td>
<td>Percentage of time the CPU was busy processing system or user requests; this metric includes CPU Stolen time.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>KPI</td>
</tr>
<tr>
<td>%Stolen</td>
<td>Usually, stolen time is the percentage of time a virtual CPU waits for a real CPU while the hypervisor is servicing another virtual processor.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>KPI</td>
</tr>
<tr>
<td>%Nice</td>
<td>Percentage of CPU time spent on low priority processes.</td>
<td>Server Visibility</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>Advanced</td>
</tr>
<tr>
<td>Metric Name</td>
<td>Description</td>
<td>Basic or Server Visibility</td>
<td>Windows</td>
<td>Linux</td>
<td>Solaris</td>
<td>AIX</td>
<td>Default Monitoring Mode</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>----------------------------</td>
<td>---------</td>
<td>-------</td>
<td>--------</td>
<td>-----</td>
<td>--------------------------</td>
</tr>
<tr>
<td>System</td>
<td>Percentage of time the CPU was busy processing kernel code.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>Percentage of time the CPU was busy processing non-kernel code.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOWait</td>
<td>Percentage of CPU time spent waiting for an I/O request.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>%Irq</td>
<td>Percentage of CPU time spent servicing hardware interrupts.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>%SoftIrq</td>
<td>Percentage of CPU time spent servicing software interrupts.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>%Busy 95th Percentile</td>
<td>The CPU %Busy percentage was at this level or lower 95% of the time.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOWait 95th Percentile</td>
<td>The CPU %Busy percentage was at this level or lower 95% of the time.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total (MB)</td>
<td>The total amount of memory.</td>
<td>Basic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Used %</td>
<td>The percentage of memory used.</td>
<td>Basic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Used (MB)</td>
<td>The amount of memory used.</td>
<td>Basic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free %</td>
<td>Percentage of free or unused memory available for processes.</td>
<td>Basic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free (MB)</td>
<td>The total amount of free or unused memory available for processes.</td>
<td>Basic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swap Total (MB)</td>
<td>The total amount of allocated swap space. Reported for each hierarchical group.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swap Used (MB)</td>
<td>The amount of swap space used. Reported for each hierarchical group.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swap Used %</td>
<td>The percentage of available swap space used. Reported for each hierarchical group.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swap Free %</td>
<td>Percentage of free swap space. Reported for each hierarchical group.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pages Paged In 95th Percentile</td>
<td>The paging-in rate to memory was at this level or lower 95% of the time.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pages Paged Out 95th Percentile</td>
<td>The paging-out rate from memory was at this level or lower 95% of the time.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pages Swapped In 95th Percentile</td>
<td>The swapping rate of pages from disks was at this level or lower 95% of the time.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pages Swapped Out 95th Percentile</td>
<td>The swapping rate of pages to disks was at this level or lower 95% of the time.</td>
<td>Server Visibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2 See Percentile Metric Reporting must be Enabled.
3 See Default Monitoring Mode.
4 See Configure Free/Used Memory Metric Calculation on Linux.

## Disk and Partition Metrics

The agent reports metrics for each disk partition and for disks in aggregate. Only mounted partitions are reported. Only local partitions are reported.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
<th>Basic or Server Visibility</th>
<th>Windows</th>
<th>Linux</th>
<th>Solaris</th>
<th>AIX</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>KB read/sec</td>
<td>The number of KB per second read from all disks and partitions.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>诊断模式</td>
</tr>
<tr>
<td>KBwritten/sec</td>
<td>The average amount of data per second written to all disks and partitions.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>诊断模式</td>
</tr>
<tr>
<td>Reads/sec</td>
<td>Number of read operations per second performed on all disks and partitions.</td>
<td>Basic</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
<td>❌</td>
<td>KPI</td>
</tr>
<tr>
<td>Writes/sec</td>
<td>Number of write operations per second performed on all disk and partitions.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>诊断模式</td>
</tr>
<tr>
<td>Avg IO Utilization (%)</td>
<td>The average time spent processing read/write requests on all disks and partitions 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.</td>
<td>Server Visibility</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
<td>KPI</td>
</tr>
<tr>
<td>Avg Service Time (ms)</td>
<td>The percentage of CPU processing consumed by a partition during read and write operations.</td>
<td>Basic</td>
<td>❌</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
<td>诊断模式</td>
</tr>
<tr>
<td>Avg Service Time (ms)</td>
<td>Time in milliseconds spent performing read and write operations across one partition.</td>
<td>Basic</td>
<td>❌</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
<td>诊断模式</td>
</tr>
<tr>
<td>Avg Queue Time (ms)</td>
<td>Time in milliseconds that a read or write request is in the queue before it gets processed across one partition.</td>
<td>Basic</td>
<td>❌</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
<td>KPI</td>
</tr>
<tr>
<td>Avg KB read/sec</td>
<td>The average amount of data per second read from one specific partition.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>诊断模式</td>
</tr>
<tr>
<td>Avg KB written/sec</td>
<td>The average amount of data per second written to one specific partition.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>诊断模式</td>
</tr>
<tr>
<td>Avg Reads/sec</td>
<td>Number of read operations per second performed on one specific partition.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>KPI</td>
</tr>
<tr>
<td>Avg Writes/sec</td>
<td>Number of write operations per second performed on one specific partition.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>KPI</td>
</tr>
<tr>
<td>Space Available</td>
<td>The amount of unused or free disk space on a specific partition in KB.</td>
<td>Basic</td>
<td>✔️</td>
<td>✔️</td>
<td>4</td>
<td>✔️</td>
<td>KPI</td>
</tr>
<tr>
<td>Space Used</td>
<td>The amount of used or unavailable disk space on a specific partition in KB.</td>
<td>Basic</td>
<td>✔️</td>
<td>4</td>
<td>✔️</td>
<td>✔️</td>
<td>高级模式</td>
</tr>
<tr>
<td>Avg IO Utilization (%)</td>
<td>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.</td>
<td>Server Visibility</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
<td>KPI</td>
</tr>
<tr>
<td>Avg read time (ms)</td>
<td>The average time required to service a read request by all disks or by one specific disk or partition.</td>
<td>Server Visibility</td>
<td>2</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>KPI</td>
</tr>
<tr>
<td>Avg write time (ms)</td>
<td>The average time required to service a write request.</td>
<td>Server Visibility</td>
<td>2</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>KPI</td>
</tr>
</tbody>
</table>
Queue Time 95th Percentile

The queue time for read/write requests was this long or shorter 95% of the time for the reported time window.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
<th>Basic or Server Visibility</th>
<th>Windows</th>
<th>Linux 1</th>
<th>Solaris 2</th>
<th>AIX</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total (MB)</td>
<td>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.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>KPI</td>
</tr>
<tr>
<td>Free (MB)</td>
<td>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.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>KPI (volume)</td>
</tr>
<tr>
<td>Used (MB)</td>
<td>The amount of storage space in use across all listed volumes or on the selected volume at the specified mount point.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>KPI (volume)</td>
</tr>
<tr>
<td>Used (%)</td>
<td>The percentage of storage space in use across all listed volumes or on the selected volume at the specified mount point.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>KPI</td>
</tr>
</tbody>
</table>

1 Only `/dev` volumes are monitored on Linux.

2 Only `/dev/dsk` and `/rpool` volumes are monitored on Solaris.

3 See Default Monitoring Mode.

Volume Metrics

AppDynamics Server Visibility gets the volume space metrics on POSIX systems using the `df` command. The volume metrics are reported across all listed volumes and for each volume at a specified mount point, such as `/boot`. Only local volumes are reported.

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. These metrics are shown as percentages in the Server Dashboard and are scaled by 100 in the Metric Browser.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
<th>Basic or Server Visibility</th>
<th>Windows</th>
<th>Linux</th>
<th>Solaris</th>
<th>AIX</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last 1 minute</td>
<td>CPU Load, presented as an average over the last 1 minute.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>KPI</td>
</tr>
<tr>
<td>Last 5 minutes</td>
<td>CPU Load, presented as an average over the last 5 minutes.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>KPI</td>
</tr>
<tr>
<td>Last 15 minutes</td>
<td>CPU Load, presented as an average over the last 15 minutes.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>KPI</td>
</tr>
</tbody>
</table>
1 See Default Monitoring Mode.

### Machine Metrics

| Metric Name | Description | Basic or Server Visibility | Windows | Linux | Solaris | AIX | Default Monitoring Mode
|-------------|-------------|---------------------------|---------|-------|--------|-----|--------------------------
| Availability | The percentage of time the Machine Agent was reporting to the Controller. In the Server Visibility UI, this provides a percentage with 6 digits of precision (i.e. 100.0000), in other words measuring up to six digits 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. | Server Visibility | ✔️ | ✔️ | ✔️ | ❌ | KPI

1 See Default Monitoring Mode.

### Network Metrics

The Standalone Machine Agent is configured to ignore virtual networks by default. For information about configuring the agent, see Machine Agent Settings for Server Visibility.

| Metric Name | Description | Basic or Server Visibility | Windows | Linux | Solaris | AIX | Default Monitoring Mode
|-------------|-------------|---------------------------|---------|-------|--------|-----|--------------------------
| Incoming KB | The volume of data received by all monitored network devices. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | Advanced
| sec         | The amount of data per second received by all monitored network devices. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | KPI
| packets     | The number of packets received by all monitored network devices. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | Advanced
| sec         | The number of data packets per second received by all monitored network devices. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | Advanced
| Outgoing KB | The volume of data sent by all monitored network devices. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | Advanced
| sec         | The volume of data sent per second by all monitored network devices. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | KPI
| packets     | The number of packets sent by all monitored network devices. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | Advanced
| packets/sec | The number of data packets sent per second by all monitored network devices. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | Advanced
| Errors/min  | The number of incoming packet errors the network incurs every minute. | Server Visibility | ✔️ | ✔️ | ✔️ | ❌ | KPI
| Errors/min  | The number of outgoing packet errors the network incurs every minute. | Server Visibility | ✔️ | ✔️ | ✔️ | ❌ | KPI
| 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 utilization might be reported as 0%. Reported for: | Server Visibility | ✔️ | ✔️ | ✔️ | ❌ | KPI
| | - Servers | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | KPI
| | - &lt;group&gt; | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | KPI
| | - &lt;network interface&gt; | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | KPI
| &lt;network interface&gt; Incoming KB | The volume of data received by the selected network interface. | Basic | ✔️ | ✔️ | ✔️ | ✔️ | Advanced
<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
<th>Basic or Server Visibility</th>
<th>Windows</th>
<th>Linux</th>
<th>Solaris</th>
<th>AIX</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>The number of processes in this class consuming CPU or memory resources or the total number of processes in this class.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Diagnostic</td>
</tr>
<tr>
<td>CPU Used (%)</td>
<td>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. If there are four cores on the machine and the process is executing four threads, each executing on one core, then the process can use up to 400% CPU. If there are four cores on the machine and the process is executing one thread on one core, then the process can use up to 25% CPU.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>KPI</td>
</tr>
<tr>
<td>Major Faults/sec</td>
<td>Number of major page faults caused by all processes in this class.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Diagnostic</td>
</tr>
<tr>
<td>Minor Faults/sec</td>
<td>Number of minor page faults caused by all processes in this class.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Diagnostic</td>
</tr>
<tr>
<td>Memory Used (%)</td>
<td>Percentage of memory consumed by the top 10 consuming processes or the percentage of memory used by all processes in this class.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>KPI</td>
</tr>
<tr>
<td>Memory Used (KB)</td>
<td>Amount of memory used by all processes in this class.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Advanced</td>
</tr>
</tbody>
</table>

1 See Default Monitoring Mode.

2 The ServerMonitoring and JavaHardwareMonitor extensions calculate unidirectional throughput (packet/KB) metrics differently. ServerMonitoring reports the total number of packets/bytes in the last minute. JavaHardwareMonitor reports the total number of packets/bytes from the time the Standalone Machine Agent started.

Process Metrics

The following metrics are aggregated and reported for each process or process class (except Total Process Count, which measures all processes observed by the agent). To modify the default process monitoring, see Machine Agent Settings for Server Visibility.
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.

Thread Count

The number of kernel threads used by all processes in this class.

Total Process Count

All individual processes observed by the Standalone Machine Agent, before it filters out unreported processes and groups the remaining processes into classes. This metric is useful for identifying anomalies in the number of processes running on a specific machine. For example: if the average Total Process Count on a machine is usually 200 to 300, and then suddenly rises to 2000 or more, this could indicate a problem on that machine.

To collect this metric, the Machine Agent captures the number of processes every 30 seconds and calculates the average number of processes per minute. This means that

- Observed is the average number of processes per minute.
- Min and Max are the 30-second buckets with the least and most processes.
- Count and Sum are the number of process captures each minute and the sum of the processes in each capture for that minute (capture1 + capture2).

Service Availability Metrics

Reported for each service that is configured.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
<th>Basic or Server Visibility</th>
<th>Windows</th>
<th>Linux</th>
<th>Solaris</th>
<th>AIX</th>
<th>Default Monitoring Mode</th>
<th>KPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Time (ms)</td>
<td>The elapsed time between sending a request and receiving a response from the monitored service.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>❌</td>
<td>KPI</td>
<td></td>
</tr>
<tr>
<td>Response Size (bytes)</td>
<td>The size of the response received from the monitored service.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>❌</td>
<td>KPI</td>
<td></td>
</tr>
<tr>
<td>Success Rate (%)</td>
<td>The percentage of successful requests over all requests made to the service.</td>
<td>Server Visibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>❌</td>
<td>KPI</td>
<td></td>
</tr>
</tbody>
</table>

Configure Disk Usage Metric Collection on Linux

You can configure the Linux Standalone Machine Agent to calculate the following metrics in a manner more consistent with the Linux `df` command:

- Servers > Volumes > /opt > Disk Usage

1 See Linux Kernel Processes.

2 When collecting Process metrics on Solaris, the Standalone Machine Agent observes and captures only the first 80 characters of each process name and argument list. This means that the agent considers only the first 80 characters of each process string when it applies whitelists ("always monitor this process") and blacklists ("never monitor this process") defined in `ServerMonitoring.yml`.

3 See Default Monitoring Mode.

Enable Percentile Metric Reporting

Percentile metric reporting must be enabled on both the Controller and the Machine Agent. By default, reporting is disabled on the Controller and enabled on the agent.

- To enable/disable reporting on the Controller, log in to the Controller administration console and set the `sim.machines.percentileMonitoringAllowed` property. See Controller Settings for Standalone Machine Agents.
- To enable/disable reporting on the agent, open the `<machine_agent_home>/extensions/ServerMonitoring/conf/ServerMonitoring.yml` file and set the `percentileEnabled` property. See Machine Agent Settings for Server Visibility.
Linux includes a mechanism for reserving some disk space to ensure that the system keeps functioning even if non-privileged users consume all other disk space. By default, these metrics do not include this reserved space. This means that the total disk space reported by the agent might differ from the total disk space reported by the Linux `df` command or other sources.

To override this default, and to ensure that these metrics include this reserved space, run the agent with the following command-line argument:

```
-Dappdynamics.machine.agent.extensions.calcVolumeFreeAndUsedWithDfCommand=true
```

This feature is supported for Linux versions of the Standalone Machine Agent only.

### Configure Free/Used Memory Metric Collection on Linux

You can configure how the Standalone Machine Agent calculates the amount of free and used memory on Linux machines. By default, the agent calculates any slab-reclaimable memory as used (not free) memory.

To configure the agent to calculate slab-reclaimable memory as free (not used) memory, run Standalone Machine Agent from the command line with the following argument:

```
-Dappdynamics.machine.agent.extensions.countSlabReclaimableAsFreeMem=true
```

This setting impacts the following metrics:

- Memory Usage (in Server Dashboard)
- Memory Used %, Memory Used MB, Memory Free %, and Memory Free MB (in Metric Browser)

### Viewing Server Visibility Metrics

Server Visibility metrics can only be viewed in the Server Visibility 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 UI.

### Standalone Machine Agent vs. .NET Machine Agent

If a server has both Machine and .NET Agents installed, there might be differences in metric values reported by the Machine Agent and the .NET Agent due to different averaging rates and measurement methods. For more information, see [.NET Compatibility Mode](#).

### Streamlined Browsing Mode

The Server Metric Browser includes a Streamlined Browsing option for node metrics. This option is useful when you are browsing metrics for tiers that contain multiple nodes.

- With Streamlined Browsing enabled, each node in the browser tree view shows only metrics reported by the Machine Agent on the corresponding node.
- With Streamlined Browsing enabled, the Hardware Resources and Custom Metrics folders are not displayed.
- With Streamlined Browsing disabled, each node in the browser tree view also shows metrics for other nodes in the same tier (with no indication of the node that corresponds to which tier group).

This streamlining option is not enabled by default. AppDynamics recommends that this option be enabled. To enable this option, go to `http://<controller_host>:<port>/controller/admin.jsp` and set `sim.metrics.metricBrowser.machineMetricMappings.enabled` to `true`.

<table>
<thead>
<tr>
<th>Streamlined Browsing Enabled</th>
<th>Streamlined Browsing Disabled</th>
</tr>
</thead>
</table>
Linux Kernel Processes

The Standalone Machine Agent has a `processSelectorRegex` setting that specifies the set of processes monitored by the agent. The default regex filters out most kernel processes on Linux machines. The agent considers monitored processes only when it calculates CPU, Memory, and Process metrics. Therefore, these metrics might be different from metrics reported by other sources such as Linux commands.

Default Monitoring Mode

The Default Monitoring Mode column indicates the default category of each metric when Dynamic Monitoring Mode (DMM) is enabled. When this mode is enabled, a Standalone Machine Agent reports metrics based on the DMM setting on that server:

- **KPI** – Report Key Performance Indicator metrics only
- **Diagnostic** – Report KPI and Diagnostic metrics
- **Advanced** – Report all unfiltered metrics on the Machine Agent.

Load Average Percentages are Scaled by 100 in the Metric Browser

Load Average metrics are shown as float values (such as 0.70 or 1.05) in the Server Dashboard. In the Metric Browser, these metrics are multiplied by 100 to provide two decimal points of precision. If a server has an average load of 7.67, for example, the Server Dashboard shows the Load Average as 7.67 and the Metric Browser shows the Load as 767. If you want to use one of the metrics in a health rule or custom dashboard, divide the metric by 100.

Stolen Times are Reported Differently

On Windows Machine Agents, if CPU %Stolen is not matching values reported by AppDynamics, it may be because Windows Performance Monitor’s, also known as Perfmon, most counters operate at a granularity of 100ns for CPU metrics. For CPU %Stolen, the counters are provided by the Hypervisor, which operates on the granularity of 1ms. Windows perfmon actually divides the CPU %Stolen counter (operating on 1ms) directly by time (in units of 100ns), due to which the CPU %Stolen values reported are very low (~0.01%). AppDynamics makes the correction between different units of time, which is why values reported by AppDynamics is greater than Perfmon’s values by a factor of 10,000 (time in 1ms * 10,000 = time in 100ns). CPU %Stolen values reported by AppDynamics sometimes exceed 100% under high load. These CPU %Stolen values are due to the multiple cores being used by the guest machine, where %Stolen time is added across multiple cores.

To compute the %Stolen time:
1. On a PowerShell terminal in the monitored guest machine, run `Get-WmiObject Win32_PerfRawData_vmGuestLib_VCPU`
2. Note the CpuStolenMs counter (in ms)
3. Note the Timestamp_PerfTime counter (in ms, take a look at Frequency_PerfTime, this value should be 1000 (in hz))
4. After a minute, run the above command again and take note of both the counters again.
5. CPU %Stolen = 100% * (CpuStolenMs (at T2) - CpuStolenMs (at T1)) / (Timestamp_PerfTime (T2) - Timestamp_PerfTime (T1))

**HardwareMonitor and JavaHardwareMonitor Calculate "Writes/Sec" Differently on AIX Machines**

The Machine Agent calculates the Disk metric **Writes/sec** on AIX machines differently, depending on whether the HardwareMonitor or the JavaHardwareMonitor extension is used. These extensions assume different block sizes when calculating these metrics. For HardwareMonitor, the block size is 4096 bytes. For JavaHardwareMonitor (based on SIGAR), the block size is 512 bytes. Because the number of writes/sec is calculated as `<kb_written>/block_size` for HardwareMonitor, the results are lower than those calculated by JavaHardwareMonitor.
Standalone Machine Agent

You use the Standalone Machine Agent (Machine Agent) to collect basic hardware metrics. The Machine Agent is a Java program that has an extensible architecture enabling you to supplement the basic metrics reported in the AppDynamics Controller UI with your own custom metrics. The Machine Agent is also the delivery vehicle for AppDynamics Server Visibility, which provides an expanded set of hardware metrics and additional monitoring capability. Server Visibility requires its own license.

Functionality provided by the Machine Agent includes:

- Reporting basic hardware metrics from the server’s OS, for example, %CPU and memory utilization, disk and network I/O
- Reporting metrics passed to the Controller by extensions
- Running remediation scripts for policy actions
- Running JVM Crash Guard

One Machine Agent license is included for each machine that has a licensed App Agent installed. The Machine Agent collects infrastructure metrics from multiple extensions and forwards them to the Controller. You can also use these metrics to find correlations between infrastructure issues on one or more servers and application-performance issues reported by the App Agents.
Standalone Machine Agent Requirements and Supported Environments

This topic lists the application environments and versions supported by the Standalone Machine Agent. Environments or versions that are not listed are not supported.

Standalone Machine Agent Supported Platforms

Supported platforms and environments for the Standalone Machine Agent are dependent on which metric data collection extension is being used. This depends on the machine's OS. See Machine Agent Metric Collection for details.

JRE Requirements

The Standalone Machine Agent requires a Java Virtual Machine. JRE 1.8 is required. Downloads for many of the supported OSs include Oracle JRE 1.8. The Standalone Machine Agent should work with most, if not all, of the JVMs supported by the Java Agent that are JRE 1.8. However, it is extensively tested only for the Oracle JDK and OpenJDK.

Note the following about the available Machine Agent downloads:

- For tested platforms listed in the following table, AppDynamics recommends using the bundled version of the 4.4 Standalone Machine Agent.
- To run the Machine Agent on other platforms (such as AIX or HP-UX), use the unbundled Machine Agent ZIP without the JRE. The agent host must have JRE 1.8 installed.
- If you are using a 64-bit Operating System, use only a 64-bit JRE.

Bash Requirements

Linux and Solaris agents require Bash 3.1 or higher.

Required Libraries

Linux systems must include the procps library.

The following table provides instructions on how to install the library on some common flavors of Linux operating system.

<table>
<thead>
<tr>
<th>Linux Flavor</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat and CentOS</td>
<td>Use yum to install the library, such as:</td>
</tr>
<tr>
<td></td>
<td>- yum install procps</td>
</tr>
<tr>
<td>Fedora</td>
<td>Install the library RPM from the Fedora website:</td>
</tr>
<tr>
<td></td>
<td>- yum install procps</td>
</tr>
</tbody>
</table>
Ubuntu

Use apt-get, such as:

- `sudo apt-get install procps`

Debian

Use a package manager such as APT to install the library (as described for the Ubuntu instructions above).

SUSE 12

Use `zypper` to install the library, such as:

- `zypper install procps`

## Supported Environments

### Tested Platforms

<table>
<thead>
<tr>
<th>OS</th>
<th>Architecture</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux / CentOS</td>
<td>Yes</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>7</td>
</tr>
<tr>
<td>Linux / Debian</td>
<td>Yes</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>9</td>
</tr>
<tr>
<td>Linux / Fedora</td>
<td>Yes</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>29</td>
</tr>
<tr>
<td>Linux / openSUSE Leap</td>
<td>Yes</td>
<td>42.3</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>15.0</td>
</tr>
<tr>
<td>Linux / Red Hat Enterprise Linux</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>7</td>
</tr>
<tr>
<td>Linux / SUSE Linux Enterprise</td>
<td>Yes</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>15</td>
</tr>
<tr>
<td>Linux / Ubuntu</td>
<td>Yes</td>
<td>14.04</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>18.04</td>
</tr>
<tr>
<td>Solaris</td>
<td>Yes</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>11</td>
</tr>
<tr>
<td>Windows</td>
<td>Yes</td>
<td>Windows Server 2012 R2</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Windows Server 2016</td>
</tr>
</tbody>
</table>
Other Platforms

Other operating systems and versions that are supported by JRE 1.8 should also work but are not fully tested by AppDynamics.

- Oracle JRE 1.8
- HP JDK/JRE 8

You should be able to run the Bundled version of the Standalone Machine Agent on machines based on Power Architecture processors, including PowerPC processors, but Server Visibility is not supported on these platforms.

Hardware and Sizing Requirements

Agent: 1 additional GB of RAM.

Controller: Although we recommend that the AppDynamics Controller be installed on a dedicated server, in some cases the Standalone Machine Agent can co-exist with the Controller on the same system. A Controller with more than 250 nodes must run on a dedicated machine. The exact number of Machine Agents that can be supported depends on whether other agents are reporting to the same controller and whether the Machine Agents have Server Visibility enabled or not (more metrics are generated under Server Visibility). For more information, see Controller System Requirements.

JVM Memory Requirements

The lightweight Standalone Machine Agent consumes minimal resources of your computer.

AppDynamics recommends the following additional Heap and PermGen space to accommodate the agent:

- Maximum heap size (-Xmx): 100 MB
- Maximum PermGen heap size (-XX:MaxPermSize): 20 MB

Extension Considerations

The exact CPU or memory overhead added by the agent can vary depending on the extensions used and whether the extensions are Java or not. Java extensions will need more heap space; the amount required depends on how the extension is coded. If you start adding extensions, increase the maximum heap space (to 256 MB or 512 MB) to be safe. You may need to increase the size of the initial memory allocation as well. Monitor the memory consumption of the agent to ensure that there are sufficient resources allocated to it. You can also enable Garbage Collection logging on the JVM to help tune the heap size for the extensions being used. There are many resources on the Internet with advice on tuning Java memory management.

Example settings for increasing the initial heap size:

Linux and Unix systems: `<machine_agent_home>/bin/machine-agent -Xms64m`

Windows: `cscript <machine_agent_home>/machine-agent.vbs -Xms64m`
# Machine Agent Metric Collection

**On this page:**
- Java Hardware Monitor
- Hardware Monitor

**Related pages:**
- Hardware Resources Metrics
- Standalone Machine Agent Requirements and Supported Environments

The Standalone Machine Agent collects hardware metrics using default extensions appropriate to specific operating systems. The following table lists the metric collection extension and other information for each supported OS. In some limited cases, you might want to change the default collection extension to a different one. The table lists the most common reasons for changing the default extension.

Supported environments, observation rates, configurability, some metric names, and definitions are dependent on the extension.

<table>
<thead>
<tr>
<th>OS</th>
<th>Default Metric Collection Extension</th>
<th>Metrics Collected</th>
<th>Reason to Change the Default Extension</th>
<th>Supported Environments</th>
</tr>
</thead>
</table>
| Microsoft Windows | JavaHardwareMonitor (if Server Visibility is disabled) ServerMonitoring (if Server Visibility is enabled) | Basic Metrics  
Server Visibility Metrics | To Customize Metrics for Virtual Disks and External Network Traffic  
The HardwareMonitoring extension is not recommended on Windows. For more information, see Monitoring Windows -- Guidelines and Important Notes. | Standalone Machine Agent Supported Environments |
| Linux       | ServerMonitoring                     | Basic Metrics  
Server Visibility Metrics | To Customize Metrics for Virtual Disks and External Network Traffic | Standalone Machine Agent Supported Environments |
| Solaris / SunOS | HardwareMonitor (if Server Visibility is disabled) ServerMonitoring (if Server Visibility is enabled) | Basic Metrics  
Server Visibility Metrics | To Customize Metrics for Virtual Disks and External Network Traffic | Standalone Machine Agent Supported Environments |
| AIX         | JavaHardwareMonitor                  | Basic Metrics  
Server Visibility Metrics | SIGAR is not supported for your OS or Linux Distribution | SIGAR |
| HP-UX       | JavaHardwareMonitor                  | Basic Metrics  
Server Visibility Metrics | SIGAR is not supported for your OS or Linux Distribution | SIGAR |
| Mac OS X    | JavaHardwareMonitor                  | Basic Metrics  
Server Visibility Metrics | SIGAR is not supported for your OS or Linux Distribution | SIGAR |
| Solaris     | HardwareMonitor (if Server Visibility is disabled) ServerMonitoring (if Server Visibility is enabled) | Basic Metrics  
Server Visibility Metrics | To Customize Metrics for Virtual Disks and External Network Traffic | Standalone Machine Agent Supported Environments |
Java Hardware Monitor

The JavaHardwareMonitor is based on SIGAR (System Information Gatherer And Reporter). This is the legacy method of collecting basic hardware metrics and is used in pre-4.1 versions and for machines running operating systems that are not supported by the ServerMonitoring extension. The following sections list scenarios where you might want to change the extension used to collect the machine metrics.

Change Scenario: Customize Metrics for Virtual Disks and External Network Traffic

By default, the Standalone Machine Agent reports metrics for network-mounted and local disks only. Additionally, only the external network traffic is aggregated (to ensure backward compatibility with previous versions of AppDynamics). For OSs that are using the ServerMonitoring extension, you can change to the JavaHardwareMonitor if you want to configure specific disks and network interfaces to be monitored. To customize the behavior of the JavaHardwareMonitor for these metrics, see Configure Metrics for Virtual Disks and External Network Traffic - JavaHardwareMonitor Extension Only.

Metric Observation Rate

The JavaHardwareMonitor, takes observations of metrics in 2 distinct ways:

- For disk and network metrics: One observation per minute. When you view the count for these metrics in the Metric Browser you see the count of 1 per minute. If you select a 15-minute interval, the count would be 15 times 1 = 15 and so on.
- For CPU and memory metrics: One observation every two seconds. When you view the count for these metrics you see the count of 30 per minute. If you select a 15-minute interval, the count would be 15 times 30 = 450 and so on.

Collect Basic Metrics Using JavaHardwareMonitor

Linux and Windows machines use the ServerMonitoring extension by default to report basic metrics. Note that, if you have Server Visibility enabled, falling back to the JavaHardwareMonitor only affects the collection of the basic hardware metrics. The extended Server Visibility metrics are still correctly collected by the ServerMonitoring extension.

To change from ServerMonitoring to JavaHardwareMonitor

1. Stop the Machine Agent.
2. Disable basic ServerMonitoring using the following steps:
   a. Edit the ServerMonitoring.yml file from <machine_agent_home>/extensions/ServerMonitoring/conf/.
   b. Change the value of basicEnabled to "false". The basicEnabled setting controls whether the Machine Agent reports basic hardware metrics through the ServerMonitoring extension. Setting this to false enables the JavaHardwareMonitor to report the basic hardware metrics using the legacy SIGAR-based reporting.
   c. Save the file.
3. Enable JavaHardwareMonitor using the following steps:
   a. Edit monitor.xml from <machine_agent_home>/monitors/JavaHardwareMonitor/.
      - For Linux, remove this line: <enable-override os-type="linux">false</enable-override>.
      - For Windows: remove this line: <enable-override os-type="windows">false</enable-override>.
   b. Save the file.
4. Restart the Machine Agent.

Hardware Monitor

The HardwareMonitor extension is a collection of OS-specific scripts. SunOS and Solaris machines use this extension by default. For the operating systems that use the JavaHardwareMonitor by default, you might want to switch to an OS-specific monitor in some cases. You can switch to the OS-specific monitors in cases where the JavaHardwareMonitor fails to report statistics and you see error logs similar to the following:
Collect Basic Hardware Metrics using HardwareMonitor

To change from JavaHardwareMonitor to an OS-specific monitor

1. Stop the Machine Agent.
2. Disable the JavaHardwareMonitor using the following steps:
   a. Locate and edit monitor.xml from `<machine_agent_home>/monitors/JavaHardwareMonitor/`
   b. Change the value `<enabled>true</enabled>` to `<enabled>false</enabled>`.
   c. Save the file.
3. Enable the HardwareMonitor for the OS you want to monitor. Use the following steps:
   a. Edit monitor.xml from `<machine_agent_home>/monitors/HardwareMonitor/`
   b. Change `<enabled>false</enabled>` to `<enabled>true</enabled>`.
   c. Save the file.
4. Restart the machine agent.

Modify Free Memory Metric Configuration

If your Machine Agent installation is using an OS-specific HardwareMonitor for metric collection, then, by default, the agent reports free memory as the memory that is not used by any process and is not in an I/O buffer or cache. It is more useful for the free memory metric to include memory that is in an I/O buffer or cache but can be available for new processes.

To include the memory in I/O buffers or cache that can be made available to new processes, modify the HardwareMonitor configuration:

1. Open `<machine_agent_home>/monitors/HardwareMonitor/config.sh`
2. Set `REPORT_MEMORY_FREE_AS_MEMORY_AVAILABLE` to 1.
3. Restart the agent.
Monitoring Windows -- Guidelines and Important Notes

On this page:
- Server Visibility on Windows
- Running the Machine Agent and .NET Agent on the Same Machine

Please note the following when using the Machine Agent on Windows.

- It is good practice to check the server frequently to make sure it has the latest Windows updates installed.
- If you have a Server Visibility license, use the ServerMonitoring extension.
- If you do not have a Server Visibility license, or if you want to collect Basic metrics only, use the JavaHardwareMonitor extension.
  - There is a known Windows issue when using the HardwareMonitor extension to collect metrics on Windows 8: The "Win32_Service" WMI class leaks memory in Windows Server 2008 R2 and in Windows 7
  - For this reason, the HardwareMonitor extension is not recommended on Windows. Use JavaHardwareMonitor instead.
  - If you must use the HardwareMonitor extension on Windows, make sure you apply the following Microsoft hotfix to the server: https://support.microsoft.com/en-us/kb/981314

Server Visibility on Windows

For additional guidelines, see Server Visibility Requirements and Supported Environments.

Running the Machine Agent and .NET Agent on the Same Machine

If you want to collect Machine or Server metrics on a machine with Machine and .NET Agents installed, you must enable .NET Compatibility Mode on both the Controller and the Machine Agent. For more information, see .NET Compatibility Mode.
Configure Metrics for Virtual Disks and External Network Traffic -
JavaHardwareMonitor Extension Only

Related pages:
- Machine Agent Metric Collection

By default, the Machine Agent reports metrics for network-mounted and local disks only. Also, only the external network traffic is aggregated (to ensure backward compatibility with previous versions of AppDynamics).

You can customize the default behavior by modifying the auto-generated configuration file, task-template.xml. The task-template.xml file provides information about the current configuration of the Machine Agent. It is created at first startup of the agent and discovers the disks and network hardware the agent can monitor on the machine where it is installed.

To customize default Machine Agent metric collection

1. Locate and edit the task-template.xml file.
   
   This file is located in the `<machine_agent_home>/monitors/JavaHardwareMonitor/` directory when the Machine Agent starts the first time.

2. Modify the task-template.xml file as follows:
   - To enable aggregation operation for localhost (lo) network metrics, change the value of the aggregate attribute (for the network element "lo") to "true".
   - To enable monitoring for a virtual disk, set the value of the enabled attribute to "true" for that disk.

A sample task-template.xml file is shown here:

```
<config>
  <disk aggregate="false" enabled="false">sunrpc</disk>
  <disk aggregate="true" enabled="true">/dev/sdb1</disk>
  <disk aggregate="false" enabled="false">proc</disk>
  <disk aggregate="false" enabled="false">none</disk>
  <disk aggregate="false" enabled="false">devpts</disk>
  <disk aggregate="false" enabled="false">/dev/sda1</disk>
  <disk aggregate="true" enabled="false">/dev/mapper/saas4-binlog</disk>
  <disk aggregate="false" enabled="false">sysfs</disk>
  <disk aggregate="false" enabled="false">tmpfs</disk>

  <network aggregate="true" enabled="true">lo</network>
  <network aggregate="true" enabled="false">sit0</network>
  <network aggregate="true" enabled="true">eth0:1</network>
  <network aggregate="true" enabled="true">eth0</network>
  <network aggregate="true" enabled="false">eth1</network>
</config>
```
3. Rename the task-template.xml file to task.xml or it will be overwritten by the Machine Agent.
4. To monitor a special device that is not enabled, add a file named "task.xml" in the `<machine_agent_home>/monitors/JavahardwareMonitor/` directory.
   The format of the task.xml file must be exactly the same format as the task-template.xml file.
   Not all disks and networks have to be listed in task.xml. If the Machine Agent finds a disk or a network that is not listed in task.xml, default properties are applied.
5. Restart the Machine Agent.
You can view basic machine metrics in the following locations. These do not require Server Visibility to be enabled.

- Metric Browser – Go to Application Infrastructure Performance > Hardware Resources
- Node page > Server tab – Go to Tiers & Nodes, select the node of interest, and go to the Server tab.
- Custom Dashboards – To add CPU metrics to your Custom Dashboard, add the metrics for any node on your target machine. Metrics collected for a node associated with the machine agent are for the server hosting that node. Any machine metrics collected will be the same for nodes that are on the same machine.

Server Visibility provides additional metrics and pages for viewing basic and server metrics. See Monitor Your Servers using Server Visibility.

Hardware Resources Metrics describes metrics collected by the Machine Agent.

Customize Default Metrics

Enable collection of metrics for virtual disks and external network traffic.

Add Custom Metrics

You can add script-based and Java-based custom monitoring extensions to collect additional metrics. For example, the ehCache monitoring extension available from AppDynamics Exchange can collect metrics to monitor Ehcache performance. The metrics appear in Application Infrastructure Performance | Custom Metrics section of the Metric Browser and can be used on custom dashboards.

You can also send metrics to the Machine Agent using the HTTP listener.

For a more complete discussion of adding custom metrics, see Extensions and Custom Metrics.
Install the Standalone Machine Agent

On this page:
- Installation Notes
- Installation Procedure

Related pages:
- Standalone Machine Agent Installation Scenarios
- Standalone Machine Agent Configuration Properties
- Upgrade the Standalone Machine Agent
- Installing the Machine Agent on a PHP Node

Installation Notes

Thoroughly test your deployment in a staging or test environment before deploying it to production.

JRE Requirements

- JRE 1.8 is required.
- In 4.2, JRE 1.8 is bundled with the OS-specific Machine Agent installation downloads.
- The Machine Agent starts within its own JVM.
- You can also use an existing JRE previously installed in your environment.

Permissions

- To avoid permission issues, install the agent as the same user who owns the Machine Agent files or as an administrator on the host machine.
- All files in the `<machine-agent-home>` installation directory should be readable by the Machine Agent.
- The user that runs the Machine Agent must have write privileges to the logging output directory and to the `/conf` directory in the agent installation directory.
- If you plan to enable the JVM Crash Guard, see JVM Crash Guard to review the required permissions.
- To create a non-root user to run the Machine Agent, see Permissions Required to Run the Machine Agent.
- Windows permissions for files and subfolders are inherited by default from the parent folder `<machine_agent_home>`. It is good practice to restrict permissions to users authorized to start, stop, and configure the Machine Agent:
  - Read and Write permissions to all files and subfolders under `<machine-agent-home>`.
  - Read, Write, and Execute permissions for the file `<machine-agent-home>/bin/machine-agent.vbs` (if running as a terminal application).
  - Start, Stop, and Restart permissions for the Machine Agent service (if running as a service). You only need admin privileges to install the service. The machine agent runs under the local system account which has extensive privileges on the local system, so there is no need to run the Machine Agent as Administrator, unless WMI access is revoked. Typically normal users do have WMI access. For additional information see: https://technet.microsoft.com/en-us/library/cc771551.aspx.
- The Windows machine agent needs Windows Script Host to be enabled.
- The Windows machine agent runs under the local system account which has extensive privileges on the local system, so there is no need to run the Machine Agent as Administrator, unless WMI access is revoked. Typically normal users do have WMI access. For additional information, see https://technet.microsoft.com/en-us/library/cc771551.aspx.

Installation Directory and Path Name Requirements

The following table lists characters that are not supported for the `<machine-agent-home>` directory name or in any directory in the path:

<table>
<thead>
<tr>
<th>Not Supported on any OS</th>
<th>Not Supported on Linux</th>
<th>Not Supported on Windows</th>
</tr>
</thead>
</table>

Copyright © AppDynamics 2012-2019
Configuration

The `controller-info.xml` file contains the properties for specifying agent to Controller communications. You will need to specify the Controller host, port, and account access key properties. See Plan the Machine Agent Configuration for more details. You might need to configure additional properties depending on your installation requirements.

Installation Procedure

- For upgrades from pre-4.3 versions, see Upgrade the Standalone Machine Agent.

The following steps are for new installations.

Follow these general steps to complete your installation:

Step 1: Plan the Machine Agent Configuration. Don't skip this step, there are a number of things to consider when configuring the Machine Agent.

Step 2: Download the Machine Agent installation package for your OS environment from the AppDynamics Downloads page and install it on the machine you want to monitor. If there is no download bundle for your OS, use the Machine Agent zip file without the JRE and use a separately downloaded JRE to run it. You need to download a separate JRE if the one already installed on the machine is a version below JRE 1.8.

- Install Using the Non-JRE Zip File - use this for OS environments other than Linux, Solaris, Windows, and Mac OS X
- Linux Install Using the RPM Package
- Linux Install Using ZIP with Bundled JRE
- Solaris Install Using ZIP with Bundled JRE
- Windows Install Using ZIP with Bundled JRE

Step 3: Verify the Standalone Machine Agent Installation

Step 4: Start the Machine Agent by executing `<machine_agent_home>/bin/machine-agent`. If you have java or system properties, you can add them to the end of the command. To see usage, enter `machine-agent -h`.
Standalone Machine Agent Installation Scenarios

On this page:

- APM Machine Agent Installation
- Independent Machine Agent Installation
- Unique Host ID Property

Related pages:

- Plan the Machine Agent Configuration
- Where to Specify Machine Agent Configuration
- Standalone Machine Agent Configuration Properties

This topic describes scenarios for new installations.

APM Machine Agent Installation

This scenario is a host server running one or more instrumented applications and one machine agent. When you install the Machine Agent on the same server with any app agent, you should not specify the application name and tier name. The one Machine Agent reports the hardware metrics to each node where the app agent and the machine agent have matching host IDs.

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.

Licensing Note
Version 4.3 and higher includes one machine agent license with each APM app agent. This machine agent can only be installed on the same server with the app agent.

Independent Machine Agent Installation

This scenario is a host server running only the machine agent and no app agents. It is not required to specify an application and tier in this case. The Machine Agent metrics can be viewed from the Servers list or the metric browser. This scenario requires a Server Visibility license.

Licensing Note
Version 4.3 and higher requires a Server Visibility license for an Independent Machine Agent Install.

If you are doing an Independent Machine Agent installation and you want the hardware metrics reported to nodes in an application, then you must configure the application name and tier name where you want the metrics to appear. The node name defaults to Node1 for the machine.

Configure the following properties using the values for the application name and tier where you want to see the metrics reported:

- application name
- tier name
- uniqueHostId for both the machine agent and app agent: use the same value

Unique Host ID Property

The uniqueHostId property is not required, however, if you don’t define a uniqueHostId the Machine Agent uses the Java API to get the host ID. The results from the API can be inconsistent and in fact, the same JVM can sometimes return a different value for the same machine each time the machine agent is restarted. To avoid problems of this nature, we recommend that you set the value of uniqueHostId to the host ID that you want to see in the UI. Use the same value for uniqueHostId for the App Agent.
Plan the Machine Agent Configuration

On this page:
- Collect Configuration Information

Related pages:
- Where to Specify Machine Agent Configuration
- Standalone Machine Agent Configuration Properties
- Configure the Standalone Machine Agent

AppDynamics provides multiple ways to configure the Machine Agent. As part of planning your installation, decide which options work best in your environment:

- controller-info.xml
- system properties
- environment variables

See Configure the Standalone Machine Agent for more details.

Collect Configuration Information

Determine your configuration requirements and collect the necessary information. For details on any of the specified configuration properties listed below, see Standalone Machine Agent Configuration Properties

<table>
<thead>
<tr>
<th>Planning Item</th>
<th>Description</th>
</tr>
</thead>
</table>
| Determine your installation scenario | For new installations, see Standalone Machine Agent Installation Scenarios  
For upgrades, see Upgrade the Standalone Machine Agent |
| Location of the startup script | Typically you add startup arguments and system properties in the startup script. If you are using a Java service wrapper, you need to know the location of the wrapper configuration. Also, see Where to Specify Machine Agent Configuration. |
| Memory and CPU | See Standalone Machine Agent Requirements and Supported Environments. |
| Connecting the agent to the Controller | You need to know the Controller host, Controller port, and account access key where you want your metrics to be reported. The account access key is generated at the Controller installation time and can be found on the AppDynamics Administration Console. |
| (Optional) SSL Communications | If you plan to use SSL to communicate with the Controller, review Enable SSL for Standalone Machine Agent. SSL-related properties include:  
- Controller SSL Enabled Property  
- Controller Keystore Password  
- Controller Keystore Filename  
- Force Default SSL Certificate Validation |
| Proxy Settings | If you plan to connect to the controller using a proxy server, you need to know the Proxy host, Proxy port and the path to the Proxy Password file. |
| Multi-tenant mode or SaaS Installations | Configure multi-tenant mode or SaaS account Information, using the Account Name property in addition to the Account Access property. This information is provided in the Welcome email from the AppDynamics Support Team. You can also find this information in the <controller_home>/initial_account_access_info.txt file. |
| Deploying from a common directory | See Deploy Multiple Machine Agents From a Common Directory. |
| Extension considerations | If you plan to use a number of extensions in your environment, review the section "Extension Considerations" in this topic: Standalone Machine Agent Requirements and Supported Environments. |
| Enabling Server Availability and Service Availability Monitoring | See [Enable Server Visibility](#). Server Availability requires a separate license. |
Install Using the Non-JRE Zip File

On this page:
- Install the Machine Agent

Related pages:
- Plan the Machine Agent Configuration
- Standalone Machine Agent Installation Scenarios
- Standalone Machine Agent Configuration Properties

This topic describes how to install the Standalone Machine Agent using the zip file that does not include the JRE. You can use this zip file if your environment already has Java 1.8 (or higher) or when installing on an OS environment that does not have an OS-specific installation package on the AppDynamics download site. The agent uses the JRE specified in the $JAVA_HOME environment variable or the default JRE for the server (enter `which java` in a terminal window).

Install the Machine Agent

1. Before installing, review the installation notes here: Install the Standalone Machine Agent.
2. Download and unzip the ZIP file: Extract the contents to the agent installation directory, `<machine_agent_home>`.
3. Gather your configuration details and configure the agent by editing `<machine_agent_home>/conf/controller-info.xml` file or by adding system properties to the JVM startup script file.
   See Plan the Machine Agent Configuration.
   a. (Required) Configure the Controller host name and port number, and account access key
   b. (Required for Multi-Tenant Mode or SaaS installations) Configure the Agent Account Information. See Controller Accounts (Multi-Tenancy).
   c. (Optional) Review memory requirements. See Standalone Machine Agent Requirements and Supported Environments
   d. (Optional) To configure the agent to use SSL see Enable SSL for Standalone Machine Agent.
   e. (Optional) To configure the agent to use proxy settings see the proxy settings in Standalone Machine Agent Configuration Properties.
   f. (Optional) For application and tier name, see Standalone Machine Agent Installation Scenarios to determine if you need to specify an application name and tier name. Usually, if you are installing the Standalone Machine Agent on the same server with any APM app agent, you should not specify application name and tier name.
4. Start the Agent
   - Windows, in a terminal window: `<machine_agent_home>/bin/machine-agent.vbs`
   - Windows as a service: `<machine_agent_home>/InstallService.vbs`
   - Linux or Unix-like: `<machine_agent_home>/bin/machine-agent`
Linux Install Using the RPM Package

On this page:
- What the RPM Package Does

Related pages:
- Standalone Machine Agent Configuration Properties
- Where to Specify Machine Agent Configuration
- Start and Stop the Standalone Machine Agent

This topic describes how to install the Standalone Machine Agent on Linux systems that support the RPM Package Manager such as CentOS, RHEL, Fedora, openSUSE, and SUSE Linux Enterprise Server. For Linux systems that do not support RPM, use the JRE Bundled Zip Archive.

The RPM installer makes the following changes to the host machine:
- Creates an appdynamics-machine-agent group and an appdynamics-machine-agent user
- Assigns ownership of certain files in the machine-agent directory to the appdynamics-machine-agent user

These changes are necessary to enable non-root users to configure and run an RPM-installed agent. If you want to use a different user or group for the machine agent service, set the MACHINE_AGENT_USER and MACHINE_AGENT_GROUP environment variables in a shell for rpm installation.

```
sudo MACHINE_AGENT_USER=myuser MACHINE_AGENT_GROUP=mygroup rpm -ivh appdynamics-machine-agent.rpm
```

If the specified user or group does not exist, an error message appears and the rpm installation is halted. To proceed you'll need to define users and groups.

If this is an issue in your environment, the alternate workflow is to install the agent using the ZIP archive (see Linux Install Using ZIP with Bundled JRE).

Install the Machine Agent

1. Before installing, review the installation notes on Install the Standalone Machine Agent.
2. Download and install the RPM Package. From the command line, enter the following where `<pkg-name.rpm>` is the name of the package for your environment, such as appdynamics-machine-agent-<version>-x86_64.rpm. This command requires administrative privileges

```
sudo rpm -ivh <pkg-name.rpm>
```

The agent files are installed in `opt/appdynamics/machine-agent` and the agent is added as a service.
3. Gather your configuration details and configure the agent by editing `<machine_agent_home>/conf/controller-info.xml` file or by adding system properties to the JVM startup script file.

See Plan the Standalone Machine Agent Configuration.

- `a. (Required)` Configure the Controller host name and port number, and account access key
- `b. (Optional)` Review memory requirements. See Standalone Machine Agent Requirements and Supported Environments
- `c. (Optional)` To configure the agent to use SSL see Enable SSL for Standalone Machine Agent.
- `d. (Optional)` To configure the agent to use proxy settings see the proxy settings in Standalone Machine Agent
Configuration Properties.

e. *(Required for Multi-Tenant Mode or SaaS installations)* Configure the Agent Account Information. See Controller Accounts (Multi-Tenancy).

f. *(Optional)* Application and tier name, see Standalone Machine Agent Installation Scenarios to determine if you need to specify an application name and tier name. Usually, if you are installing the Standalone Machine Agent on the same server with any APM app agent, you should not specify application name and tier name.

4. Start the Agent. You can start the agent as a service (requires sudo or root user) or from the command line. If systemd was detected during installation, you can use the systemctl command to start the agent service.

- Using SysV – service appdynamics-machine-agent start
- Using systemd – systemctl start appdynamics-machine-agent
- From the Command Line – <machine_agent_home>/bin/machine-agent
- From the Launcher – <machine_agent_home>/etc/init.d/appdynamics-machine-agent start

What the RPM Package Does

The RPM package manager installs the agent files to opt/appdynamics/machine-agent, creates symbolic links, sets environment variables and adds the agent as a service.

Symbolic Links

The RPM package manager creates the symbolic links to agent scripts and configuration files in the /etc directory. You can edit these links if you need to accommodate a different installation directory, JRE, or system user account running the agent.

**Link for SysV Service Script**

/etc/init.d/appdynamics-machine-agent >>
/opt/appdynamics/machine-agent/etc/init.d/appdynamics-machine-agent

This script provides the following service commands:

- Start – Starts the service, runs a script that starts the agent
- Stop – Stops the service
- Restart – Restarts the service
- Status – Returns the status of the service.

Example command: service appdynamics-machine-agent start

**Link for Environment Variables**

SysV – /etc/sysconfig/appdynamics-machine-agent >>
/opt/appdynamics/machine-agent/etc/sysconfig/appdynamics-machine-agent

This script sets up the environment variables:

- MACHINE_AGENT_HOME=/opt/appdynamics/machine-agent - Specifies where the agent files are located.
- JAVA_HOME=/opt/appdynamics/machine-agent/jre - Specifies the JRE the agent uses. The RPM package installer contains its own JRE, which we recommend you use, but you can use a different JRE as long as it meets the JRE requirements.
- MACHINE_AGENT_USER=root - Specifies the system user for starting the agent (the default is root). It is good practice to create a non-root user to run the machine agent. The new user needs to have read-access to controller-info.xml and write access to the log file. See Permissions Required to Run the Machine Agent for more details.

systemd – The environment variables are in the service file:

/etc/systemd/system/appdynamics-machine-agent.service >>
/opt/appdynamics/machine-agent/etc/systemd/system/appdynamics-machine-agent.service

"MACHINE_AGENT_USER" variable doesn’t exist in the systemd service file. The variable is just "User".

**Link for Agent to Controller Communication**
Sets up a link to the controller-info.xml file containing the properties for agent-to-controller communication.

```
/etc/appdynamics/machine-agent/controller-info.xml >>
/opt/appdynamics/machine-agent/conf/controller-info.xml
```

**Logging Configuration**

The log4j.xml file controls the detail of information logged by the agent. By default, the logging level is set to "info".

```
/etc/appdynamics/machine-agent/logging/log4j.xml >>
/opt/appdynamics/machine-agent/conf/logging/log4j.xml
```

**Adds the Agent as a Service**

After the package is installed, the appdynamics-machine-agent runs `opt/appdynamics/machine-agent/bin/postInstall.sh` to add the agent to the services. The installer either copies the SysV script to add the service or, if systemd is detected, the installer copies the agent unit file. `/etc/systemd/system/appdynamics-machine-agent.service` and adds the agent to the services using this unit file.

**Installs a systemd Unit File for the Agent Service**

If systemd is detected on the system when you install the RPM package or run the `postinstall.sh` script, the systemd unit file for the agent service is copied to the following location: `/etc/systemd/system/appdynamics-machine-agent.service`

```
ExecStart: This option in the systemd service file points to a script that starts the agent as a daemon. If you did not install the machine agent in `/opt/appdynamics/machine-agent`, then change the path to `<machine-agent-home>/scripts/machine-agent-daemon`. If you didn't change the moving the home directory it points to the correct place by default.
```

You can start and stop the agent service using the relevant systemctl commands. For information on all systemctl commands, see the man pages for `systemctl`.

<table>
<thead>
<tr>
<th>Command</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td><code>systemctl &lt;command&gt; appdynamics-machine-agent.service</code></td>
</tr>
<tr>
<td>stop</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td></td>
</tr>
<tr>
<td>restart</td>
<td></td>
</tr>
<tr>
<td>disable</td>
<td><code>systemctl &lt;command&gt; /etc/systemd/system/appdynamics-machine-agent.service</code></td>
</tr>
<tr>
<td>enable</td>
<td></td>
</tr>
<tr>
<td>*use the full path</td>
<td></td>
</tr>
</tbody>
</table>
This topic describes how to install the Standalone Machine Agent using the Linux ZIP archive that includes JRE 1.8. If your Linux distribution supports the RPM Package Manager, it is easier to download and use the RPM package.

Install the Agent

1. Before installing, review the installation notes at Install the Standalone Machine Agent.

   You need read/write privileges to the &lt;machine-agent-home&gt; installation directory.

2. Download and unzip the Agent zip bundle. On the command line, use the following command, where &lt;zip-bundle.zip&gt; is the name of the zip archive for your environment and &lt;agent_home&gt; is the name of the directory where you want to install the agent.

   `unzip &lt;zip-bundle.zip&gt; -d &lt;machine_agent_home&gt;

3. Gather your configuration details and configure the agent by editing&lt;machine_agent_home&gt;/conf/controller-info.xml file or by adding system properties to the JVM startup script file. See Plan the Standalone Machine Agent Configuration.

   a. (Required) Configure the Controller host name and port number, and account access key.
   b. (Optional) Review memory requirements. See Standalone Machine Agent Requirements and Supported Environments.
   c. (Optional) To configure the agent to use SSL see Enable SSL for Standalone Machine Agent.
   d. (Optional) To configure the agent to use proxy settings see the proxy settings in Standalone Machine Agent Configuration Properties.
   e. (Required for Multi-Tenant Mode or SaaS installations) Configure the Agent Account Information. See Controller Accounts (Multi-Tenancy).
   f. (Optional) Application and tier name. Usually, if you are installing the Standalone Machine Agent on the same server with any APM app agent, you should not specify application name and tier name. See Standalone Machine Agents and Applications, to determine if you need to specify an application name and tier name.

4. If you prefer to add the machine agent as a service, see the section Add the Agent as a Service below and start it accordingly.

5. Start the machine agent: &lt;machine_agent_home&gt;/bin/machine-agent. For example, from the &lt;machine_agent_home&gt; directory:

   `./bin/machine-agent`
To see usage, enter: `machine-agent -h`

```
       [-X prop]

Start the machine agent.
-d            daemonize (run in background)
-p pidfile    write PID to <pidfile>
-h
--help       print command line options
-D prop       set JAVA system property
-X prop       set non-standard JAVA system property
```

6. (Optional) Verify the installation. See Verify the Standalone Machine Agent Installation.

Add the Agent as a Service Using SysV

1. Install the Standalone Machine Agent as described above.
2. Create a link to `/etc/sysconfig`:

   ```
   ln -s <machine-agent-home>/etc/sysconfig/appdynamics-machine-agent 
   /etc/sysconfig/appdynamics-machine-agent
   ```

3. Copy the machine agent to `/etc/init.d`, for example:

   ```
   cp <machine-agent-home>/etc/init.d/appdynamics-machine-agent 
   /etc/init.d/appdynamics-machine-agent
   ```

4. Edit the environment variables in `/etc/sysconfig/appdynamics-machine-agent` configuration file:
   - `MACHINE_AGENT_HOME` – Specifies where the Machine Agent files are located.
   - `JAVA_HOME` – Specifies the JRE the agent uses. The Machine Agent is bundled with the JRE that we recommend you use, but you can use an existing JRE (1.8 or higher)
   - `MACHINE_AGENT_USER` – Specifies the system user which is used to start the Standalone Machine Agent. By default this is root. AppDynamics recommends that you create a non-root user to run the machine agent. The new user needs to have read-access to controller-info.xml and write access to the log file. See Permissions Required to Run the Machine Agent for details on the required execute permissions.
5. Add the agent as a service. For example, on the command line enter:

   ```
   chkconfig --add appdynamics-machine-agent
   ```
• For Ubuntu, you can use `update-rc.d` or `sysv-rc-conf` instead. Refer to Ubuntu documentation for the proper usage.


7. Verify that the Agent is reporting to the Controller. See [Verify the Standalone Machine Agent Installation](#).

Add the Agent as a Service Using systemd

1. Install the Standalone Machine Agent as described above.
2. Edit the environment variables and options in the following service file as needed: `<machine-agent-home>/etc/systemd/system/appdynamics-machine-agent.service`
   a. `MACHINE_AGENT_HOME` – Specifies where the Machine Agent files are located.
   b. `JAVA_HOME` – Specifies the JRE the agent uses. The Machine Agent is bundled with the JRE that we recommend you use, but you can use an existing JRE. (1.7 or higher for versions up to 4.3, 1.8 or higher for version 4.4 and higher)
   c. `User` – This option in the service file specifies the system user to be used to start the Standalone Machine Agent. By default this is root. AppDynamics recommends that you create a non-root user to run the machine agent. The new user needs to have read access to `controller-info.xml` and write access to the log files. See [Permissions for Non-Root User to Run the Machine Agent](#) for more details. The systemd service file does not include the "MACHINE_AGENT_USER" variable.

3. Copy the file as follows:
   ```
   cp <machine-agent-home>/etc/systemd/system/appdynamics-machine-agent.service \
   /etc/systemd/system/appdynamics-machine-agent.service
   ```

4. Enable the Machine Agent to start at system startup:
   ```
   systemctl enable appdynamics-machine-agent
   ```

5. Start the agent service:
   ```
   systemctl start appdynamics-machine-agent
   ```

6. Check the service status:
   ```
   systemctl status appdynamics-machine-agent
   ```

7. Verify that the Agent is reporting to the Controller. See [Verify the Standalone Machine Agent Installation](#).
### Solaris Install Using ZIP with Bundled JRE

This topic describes how to install the Machine Agent on 64-bit Solaris using the ZIP archive that includes JRE 1.8. Adding the Machine Agent as a service is not supported on Solaris.

#### Install the Agent

1. Before installing, review the installation notes here: Install the Standalone Machine Agent.
2. Machine agent on Solaris is supported only when the installation is done in Global zones.
3. Verify that the Solaris host has the `nawk` (new awk) utility installed. The Solaris agent requires this utility.
4. Download the Machine Agent Bundle for Solaris from the AppDynamics Download Center. There are two installation bundles available:
   - For Sparc machines: Machine - Agent Bundle - 64-bit solaris-sparcv9.zip
   - For x86 machine: Machine Agent Bundle - 64-bit Solaris.zip
5. Unzip the agent ZIP bundle as follows where `<machine_agent_home>` is the install directory.

   ```bash
   unzip <zip-bundle.zip> -d <machine_agent_home>
   ```

6. Gather your configuration details and configure the agent by editing `<machine_agent_home>/conf/controller-info.xml` file or by adding system properties to the JVM startup script file. See **Plan the Machine Agent Configuration**.
   a. *(Required)* Configure the Controller host name and port number, and account access key
   b. *(Required for Multi-Tenant Mode or SaaS installations)* Configure the Agent Account Information. See **Controller Accounts (Multi-Tenancy)**.
   c. *(Optional)* Review memory requirements. See **Standalone Machine Agent Requirements and Supported Environments**.
   d. *(Optional)* To configure the agent to use SSL see **Enable SSL for Standalone Machine Agent**.
   e. *(Optional)* To configure the agent to use proxy settings see the proxy settings in **Standalone Machine Agent Configuration Properties**.
   f. *(Optional)* Application and tier name, see **Standalone Machine Agent Installation Scenarios** to determine if you need to specify an application name and tier name. Usually, if you are installing the Standalone Machine Agent on the same server with any APM app agent, you should not specify application name and tier name.
7. Start the agent from the command line:

   ```bash
   % <machine_agent_home>/bin/machine-agent
   ```

8. Verify that the Agent is reporting to the Controller.
Windows Install Using ZIP with Bundled JRE

1. Before installing, review the installation notes: Install the Standalone Machine Agent.
2. Install all available Windows updates.
3. If your machine hasn't been restarted recently, restart your machine. Failure to do so may cause your machine to experience a CPU spike when you start the Machine Agent.
4. Download and unzip the Windows ZIP bundle: Extract the contents to the agent installation directory, `<machine_agent_home>`.
5. Gather your configuration details and configure the agent by editing `<machine_agent_home>/conf/controller-info.xml` file or by adding system properties to the JVM startup script file.

**Related pages:**
- Standalone Machine Agent Configuration Properties
- Standalone Machine Agent Installation Scenarios
- Start and Stop the Standalone Machine Agent
- Upgrade the Standalone Machine Agent
- Uninstall the Standalone Machine Agent

**Plan the Machine Agent Configuration**

- **(Required)** Configure the Controller host name and port number, and account access key.
- **(Required for Multi-Tenant Mode or SaaS installations)** Configure the Agent Account Information. See Controller Accounts (Multi-Tenancy).
- **(Optional)** To configure the agent to use SSL see Enable SSL for Standalone Machine Agent.
- **(Optional)** To configure the agent to use proxy settings see Standalone Machine Agent Configuration Properties.
- **(Optional)** Application and tier name, see Standalone Machine Agent Installation Scenarios to determine if you need to specify an application name and tier name. Usually, if you are installing the Standalone Machine Agent on the same server with any APM app agent, you should not specify application name and tier name.

6. Start the agent. If no `-D` options are needed on the command line, you can execute the `.vbs` scripts below by double-clicking on them in Windows Explorer.
   - **(As a Windows Service. You need admin privileges for this step.** You can specify AppDynamics-specific `-D` options on this command line or edit the controller-info.xml. Add other JVM properties on the command line as needed.
   - In a terminal window, execute the following command:
     ```
cscript <machine_agent_home>\InstallService.vbs <jvm_options>
```
   - Now you can start or stop the Appdynamics Machine Agent service from Windows Services.
   - **(Optional)** Application and tier name, see Standalone Machine Agent Installation Scenarios to determine if you need to specify an application name and tier name. Usually, if you are installing the Standalone Machine Agent on the same server with any APM app agent, you should not specify application name and tier name.

   a. As a Windows Service. You need admin privileges for this step. You can specify AppDynamics-specific `-D` options on this command line or edit the controller-info.xml. Add other JVM properties on the command line as needed.
   - In a terminal window, execute the following command:
     ```
cscript <machine_agent_home>\InstallService.vbs <jvm_options>
```
   - Now you can start or stop the Appdynamics Machine Agent service from Windows Services.
   b. As an application in the terminal window, execute:
     ```
cscript <machine_agent_home>\bin\machine-agent.vbs.
```

**Usage parameters for machine-agent.vbs:**
Using machine-agent.vbs

> machine-agent.vbs -h
Usage: machine-agent.vbs [-h] [-j JAVA_HOME] [-Dprop1 ...]
[-Xprop2 ...]
Start the machine agent.
    -h              print command line options
    -j JAVA_HOME    set java home for the agent
    -Dprop1         set standard system properties for the agent
    -Xprop2         set non-standard system properties for the agent

Example of machine-agent.vbs:

> cscript machine-agent.vbs
-Dappdynamics.controller.hostName=192.168.1.20
-Dappdynamics.controller.port=8090
Deploy Multiple Machine Agents From a Common Directory

You can deploy the Standalone Machine Agent on multiple hosts from one central directory. This setup can make it much easier to update the agent because each agent instance starts from one set of executables in one central location.

Workflow Description

The following steps describe the workflow:

1. Set up the central agent directory:
   a. Install the Machine Agent executables and config files in a central directory, such as an NFS-mounted shared directory
   b. Specify properties common to all remote agents in this central directory.
2. Configure each remote host to start an instance of the agent from this central directory:
   a. Specify properties that are unique to the agent on that host, either in a startup script or on the command line.
   b. Configure a startup script to start an instance of the machine agent from the central agent directory.

Set up the Central Agent Directory

1. Download the latest Machine Agent ZIP file. For more information, see Download AppDynamics Software.
2. Unzip the downloaded file to the desired root directory (`<central-agent-root-directory>`) on the central host.
3. Specify the properties common to all Standalone Machine Agent instances running on all remote hosts. Examples of common properties include the account name and the Controller host and port. You specify these properties in `<central-agent-root-directory>/conf/controller-info.xml`.

Configure Each Remote Host to Start an Agent Instance

On each Machine Agent host, specify the properties unique to that agent such as the runtime directory, application name, tier name, and node name. To specify these properties, configure `-D` parameters in the startup script or on the command line. For example, the startup script on a remote host might include the following command:

```bash
<central_agent_root_directory>/bin/machine-agent
-Dappdynamics.agent.runtime.dir=<agent_runtime_dir>
-Dappdynamics.agent.applicationName=<application_name>
```

It is good practice to specify a unique runtime directory for each remote host, to ensure that log files for each agent instance get saved to a separate directory. The agent log directory is located under the agent runtime directory, which is specified using the `-Dappdynamics.agent.runtime.dir` system property. The log directory is directly under the runtime directory (`<agent-runtime-dir>/logs`). Note the following:

- You should specify the runtime directory as an absolute path in the startup script on the remote host, like this:
To change the log directory and file name from `logs/machine-agent.log` (the default), edit the file `<machine-agent-home>/conf/logging/log4.xml`. Then change the value of the "FileAppender" > "File" parameter to the desired directory and file name.

The file-system permissions must allow creation of the log directory and file. If a failure occurs trying to create the log file, the Machine Agent uses the default file `<central-agent-root-directory>/logs/machine-agent.log`, logs the error and continues.
Verify the Standalone Machine Agent Installation

On this page:
- Check Agent Logs
- Verify that the Agent is Reporting to the Controller
- Machine Agent Not Reporting

Related pages:
- FAQs and Troubleshooting
- Standalone Machine Agent Configuration Properties

Check Agent Logs

After a successful install, the agent log at `<machine_agent_home>/logs`, should contain the following message:

```
Started AppDynamics Machine Agent Successfully
```

If the agent log file is not present, the Machine Agent may not be accessing the command properties. To troubleshoot, check the application server log file where STDOUT is logged. It has the fallback log messages, which are useful for troubleshooting the agent.

Verify that the Agent is Reporting to the Controller

1. Log in to the Controller UI as an Administrator.
2. From the Controller Top Navigation Bar, choose Settings > AppDynamics Agents.
3. Click the Machine Agents tab.
   The listing for the machine agent appears. If you don’t see the machine agent listed, check your property settings in `<machine_agent_home>/conf/controller-info.xml`. See Resolve Standalone Machine Agent Installation Problems.

Machine Agent Not Reporting

If it appears that the Machine Agent is not reporting to the Controller, check this article on the Community Knowledge Base: Machine Agent Not Reporting.
Resolve Standalone Machine Agent Installation Problems

On this page:
- Verify that the Agent is Running
- Resolve Agent Connectivity Problems

Related pages:
- FAQs and Troubleshooting
- Standalone Machine Agent Configuration Properties

Verify that the Agent is Running

Use the following command to verify that the agent process is running:

**Linux:**

```
ps -ef | grep machine
```

**Windows:**

1. Open a command line console.
2. Start the Task Manager and click the Processes tab.
3. The agent process should be running. If it is not running, stop and then restart the agent.

Resolve Agent Connectivity Problems

Make sure you have configured the Controller IP address, Controller port number, and Account Access Key in the agent startup command, script or plist, or in the controller-info.xml file.

After configuring, restart the agent and check the behavior. Standalone Machine Agent log files may also provide some insight into problems.

If when you start the Standalone Machine Agent, it cannot register with the controller or associate with the same node in the Controller, the stack trace may reveal the reason why.

For example, the following message in the stack trace may indicate that the application, tier, and node information was not provided during the Standalone Machine Agent startup command or in the controller-info.xml file.

```
<execution-output>System agent 239590 not associated with application, metric registration request refused.</execution-output>
```
Start and Stop the Standalone Machine Agent

On this page:
- Start the Standalone Machine Agent
- Stop the Standalone Machine Agent

Related pages:
- Install the Standalone Machine Agent
- Configure the Standalone Machine Agent
- Installing the Machine Agent on a PHP Node

This topic contains a summary of the start and stop commands for the Machine Agent. For information about required permissions, see Permissions Required to Run the Machine Agent.

Running the Standalone Machine Agent with a Non-Bundled JRE
If you want to run the Machine Agent with the default JRE on the machine (rather than the JRE bundled with the agent), the recommended workflow is to install a non-JRE version of the Standalone Machine Agent and run the non-JRE agent instead. For more information, see Install Using the Non-JRE Zip File.

Start the Standalone Machine Agent

RPM-based Linux Systems
If you installed using the Linux RPM, start the Machine Agent as follows:

- **SysV systems**: service appdynamics-machine-agent start
- **systemd systems**: Use the Machine Agent service launcher as follows: systemctl start appdynamics-machine-agent

Linux and Solaris Systems
If you installed using the JRE bundled zip archive for Linux and Solaris, run the launcher as follows:

- **Linux**: `<machine_agent_home>/bin/machine-agent -d -p <machine_agent_home>/pidfile`
- **Solaris**: `<machine_agent_home>/bin/machine-agent -d -p <machine_agent_home>/pidfile`

Windows Systems
See Windows Install Using ZIP with Bundled JRE.

Stop the Standalone Machine Agent

Non-JRE Bundled Zip Archive
If the Machine Agent process is running in the background, you can stop it by simply entering the kill command with the process ID as the argument. If it is running in the foreground in a console, you can press Ctrl+C to shut down the agent.

RPM-based Linux Systems
- **SysV**: Use the service launcher as follows:
% service /etc/init.d/appdynamics-machine-agent stop

**systemd:** Use the systemd unit file as follows:

% systemctl stop appdynamics-machine-agent

---

**Linux and Solaris systems**

*Running in the Foreground:* Press Ctrl-c to stop the agent process.

*Running in the Background:* Identify the process id and use the kill command.

1. **Identify the agent process:** ps -ef | grep machineagent.
   The output provides the process ID (PID) of the Machine Agent process.
2. **Stop the agent process:** kill <machine_agent_PID>.

---

**Windows**

*In the foreground in a console window:* Use Ctrl+C to shut down the agent.

*As a Windows Service:* Stop the service using either of the following methods:

- In the Windows Services application, select **AppDynamics Machine Agent** and click **Stop**.
- Use the MachineAgentService command as follows:

```
C:\> <machine_agent_home>\bin\MachineAgentService.exe /stop
```

---

**Mac OS X**

*As a background process:* enter the kill command with the process ID as the argument.

*As a service:* From the bash command line:

```
> sudo launchctl unload -w
<machine_agent_home>/com.appdynamics.machineagent.plist
```
Permissions Required to Run the Machine Agent

On this page:

- Important Notes
- Linux
- Windows
- Mac OS X, AIX, HP-UX, and Z/OS
- Solaris System Utilities
- JVM Crash Guard

Related pages:

- Install the Standalone Machine Agent

This topic describes the permissions needed to run the Machine Agent. During installation, the default user for running the Machine Agent is set to root. This is because the only user that is standard on a UNIX or Linux system is root and we do not want to create users on your system. We recommend that you create a non-root user, for example `<machine_agent_user>`, and assign the appropriate permissions to that user.

For all environments, you should create a specific user with the necessary read/write/execute permissions.

- All files in the `<machine-agent-home>` installation directory should be readable by the Machine Agent.
- The user that runs the Machine Agent must have write privileges to the logging output directory and to the `/conf` directory in the agent installation directory.
- In addition, the user that runs the Machine Agent needs execute access as described below.

Important Notes

- You do not need to run the Machine Agent from a root or administrator account, but if you enable the JVM Crash Guard on a monitored application running from a root or administrator account, the Machine Agent requires root or administrator privileges to look in the monitored application's JVM process and directory listings for crash files.
- You also will need to run with administrator or root privileges if you want to monitor networks or disks that are only available to the administrator or root user.
- The user that runs the Machine Agent must have write privileges to the `conf` and `logs` directories in the `<machine_agent_home>` directory.
- The Machine Agent implements a shutdown hook, so issuing the kill command (or Ctrl+C) from the operating system causes the agent to perform a graceful shutdown.

Linux

- ip
- df
- awk
- basename
- cat
- date
- dmesg
- md5sum
- readlink
- sed
- uname
- ps

Windows

Windows permissions for files and subfolders are inherited by default from the parent folder (`<machine_agent_home>`). It is good practice to restrict permissions to users authorized to start, stop, and configure the Machine Agent:
Read and Write permissions to all files and subfolders under `<machine-agent-home>`

Read, Write, and Execute permissions for the file `<machine-agent-home>\bin\machine-agent.vbs` (if running as a terminal application)

Start, Stop, and Restart permissions for the Machine Agent service (if running as a service). You only need admin privileges to install the service. The machine agent runs under the local system account which has extensive privileges on the local system, so there is no need to run the Machine Agent as Administrator, unless WMI access is revoked. Normal users typically have WMI access. For additional information, see https://technet.microsoft.com/en-us/library/cc771551.aspx.

The machine agent runs under the local system account which has extensive privileges on the local system, so there is no need to run the Machine Agent as Administrator, unless WMI access is revoked. Normal users typically have WMI access. For additional information, see https://technet.microsoft.com/en-us/library/cc771551.aspx.

Mac OS X, AIX, HP-UX, and Z/OS

There are no particular execute privileges needed.

Solaris System Utilities

- `awk`
- `netstat`
- `zpool`
- `egrep`
- `iostat`
- `prtconf`
- `pagesize`
- `kstat`
- `prstat`
- `grep`
- `vmstat`

JVM Crash Guard

If you plan to enable JVM Crash Guard, see JVM Crash Guard for information on additional permissions that are required.
Configure the Standalone Machine Agent

On this page:
- How to Configure the Machine Agent Properties
- Example Configuration

Related pages:
- Standalone Machine Agent Configuration Properties
- Plan the Machine Agent Configuration
- .NET Compatibility Mode

AppDynamics provides flexibility for configuring the machine agent so that you can choose the best fit for your deployment environment.

Not all options are available for all properties. See Standalone Machine Agent Configuration Properties for details.

How to Configure the Machine Agent Properties

You can configure the Machine Agent in the following ways:

- Edit the controller-info.xml file located in the `<machine_agent_home>/conf` directory
- Add agent system properties `-D<system_property>` section of the Machine Agent start-up script or on the command line
- Use environment variables. To configure the agent with environment variables, set the value of the environment variable in the environment where the monitored application runs and restart the agent. Environment variables exist for most of the agent settings in the controller-info.xml file, but not all settings are configurable through environment variables. For those settings, you need to use system properties or controller-info.xml, as described in the Standalone Machine Agent Configuration Properties.

The JVM system properties and environment variables override the settings in the controller-info.xml file. The Machine Agent applies configurations from the following sources in order. The agent applies the first non-empty value for a configuration property.

1. Environment variables.
2. System properties passed in the start command for the JVM.

For details for each OS environment and install package, see Where to Specify Machine Agent Configuration.

Example Configuration

Machine Agent controller-info.xml File
```xml
<?xml version="1.0" encoding="UTF-8"?>
<controller-info>
    <controller-host>192.10.10.10</controller-host>
    <controller-port>8090</controller-port>

    <account-access-key>165e65645-95c1-40e3-9576-6a1424de9625</account-access-key>
    <controller-ssl-enabled>false</controller-ssl-enabled>
    <enable-orchestration>false</enable-orchestration>
    <sim-enabled>false</sim-enabled>
    <unique-host-id>your-host-id</unique-host-id>
    <!-- The following account-related parameter is necessary only for SaaS installations-->
    <!--account-name></account-name-->

</controller-info>
```

**bash command-line example**

```bash
<machine_agent_home>/bin/machine-agent
-Dappdynamics.controller.hostName=192.10.10.10
-Dappdynamics.controller.port=8090
-Dappdynamics.agent.accountAccessKey=165e65645-95c1-40e3-9576-6a1424de9625
-Dappdynamics.agent.uniqueHostId=your-host-id
```
Where to Specify Machine Agent Configuration

On this page:

- Linux As a Service
- Any UNIX-like system
- Windows
- Mac OS X

Related pages:

- Install the Standalone Machine Agent
- Plan the Machine Agent Configuration
- Standalone Machine Agent Configuration Properties

You configure agent system properties in different ways depending on your operating system and whether you are starting the agent from the command line or as a service.

Linux As a Service

SysV service launcher: specify the agent system properties in the `<machine_agent_home>/etc/sysconfig/appdynamics-machine-agent` configuration file:

- Edit the JAVA_OPTS environment variable

systemd: specify the agent system properties in the `<machine_agent_home>/etc/systemd/system/appdynamics-machine-agent.service` file:

- Edit the JAVA_OPTS environment variable

Any UNIX-like system

When starting the agent application on the command line for an UNIX-like system ((Linux, Solaris, Mac, AIX, and so on) using the machine-agent command, specify the agent system properties on the command line, such as:

- Run agent in the background:
  
  ```
  nohup <machine_agent_home>/bin/machine-agent -D<system_property1>=<value1> -D<system_property2>=<value2> ... &
  ```

- Run agent in the foreground:
  
  ```
  <machine_agent_home>/bin/machine-agent -D<system_property1>=<value1> -D<system_property2>=<value2> ...
  ```

Windows

When installing and starting a Windows service (you need admin privileges), specify AppDynamics-specific -D options on the command line or in controller-info.xml. Add other JVM properties on the command line.

- `<machine_agent_home>\cscript InstallService.vbs InstallService.cmd <jvm_options>`

Mac OS X

Start the agent service using the `<machine_agent_home>/osx-install.sh` script and specify the agent system properties on the command line, such as:

```
> sh <machine_agent_home>/osx-install.sh -D<system_property1>=<value1> -D<system_property2>=<value2> ...
```

When you run the `<machine_agent_home>/osx-install.sh` script, the `<machine_agent_home>/com.appdynamics.machineagent.plist.template` is updated with the installation directory and the java properties set for the machine agent.
Standalone Machine Agent Configuration Properties

On this page:
- System Property Syntax
- Reference
  - .NET Compatibility Mode
  - Account Access Key
  - Account Name
  - Agent Runtime Directory
  - Container Process Selector Blacklist Regex
  - Controller Host
  - Controller Keystore Filename
  - Controller Keystore Password
  - Controller Port
  - Controller SSL Enabled
  - Create Node if Absent
  - Enable Docker Visibility
  - Enable HTTP Listener
  - Enable Orchestration
  - Force Default SSL Certificate Validation
  - Enable Dynamic Monitoring Mode (DMM)
  - HTTP Listener Port
  - Machine Hierarchy
  - Proxy Host
  - Proxy Password File
  - Proxy Port
  - Proxy User Name
  - Server Visibility Enabled
  - Service Availability Update Interval
  - Unique Host ID
  - Use Simple Hostname
- Independent Standalone Machine Agent Install Scenario
  - Application Name
  - Node Name
  - Tier Name

Related pages:
- FAQs and Troubleshooting
- Plan the Machine Agent Configuration
- Install the Standalone Machine Agent
- Standalone Machine Agent Installation Scenarios
- Encrypt Agent Credentials

This topic describes the agent configuration properties, including controller-info.xml elements, system property options, and environment variables where applicable. You can configure agent system properties in different ways depending on your operating system and the installation package you use. You can configure many properties using the controller-info.xml file, system properties (on the command line or in the startup script), and environment variables. The agent updates dynamically in response to agent configuration property changes, so you do not need to restart the agent.

System Property Syntax

- System properties are case-sensitive.
- Values that contain spaces must be surrounded with double-quotes.

Reference

.NET Compatibility Mode
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.

**Element in controller-info.xml:** `<dotnet-compatibility-mode>`

**System Property:** `-Dappdynamics.machine.agent.dotnetCompatibilityMode`

**Environment Variable:** N/A

**Type:** boolean

**Default:** false

**Required:** This mode is required if you want to collect and view Machine or Server metrics on a server with Machine and .NET Agents installed.

**Account Access Key**

The account access key used to authenticate with the Controller. This key is generated at installation time and can be found by viewing the license information in the Controller Settings. See License Management.

**Element in controller-info.xml:** `<account-access-key>`

**System Property:** `-Dappdynamics.agent.accountAccessKey`

**Environment Variable:** `APPDYNAMICS_AGENT_ACCOUNT_ACCESS_KEY`

**Type:** String

**Default:** None

**Required:** Prior to version 4.1, this property was required only for SaaS and multi-tenant Controllers. For versions 4.1 and higher, the account access key property is required to authenticate all agent to Controller communications.

**Example:** `-Dappdynamics.agent.accountAccessKey=165e65645-95c1-40e3-9576-6a1424de9625`

**Account Name**

The account name used to authenticate with the Controller. If you are using the AppDynamics SaaS Controller, the Account Name is provided in the Welcome email sent by AppDynamics.

**Element in controller-info.xml:** `<account-name>`

**System Property:** `-Dappdynamics.agent.accountName`

**Environment Variable:** `APPDYNAMICS_AGENT_ACCOUNT_NAME`

**Type:** String

**Default:** None

**Required:** Yes for AppDynamics SaaS Controller and multi-tenant users; not for single-tenant mode (the default). When the agent is registered with an AppDynamics SaaS Controller, features used to run Remediation Scripts are disabled if you later reconfigure the agent to register with a non-SaaS or on-premises Controller, the agent can run local scripts as usual.

**Agent Runtime Directory**

Sets the runtime directory for all runtime files, such as logs, for nodes that use this agent installation. If this property is specified, all agent logs are written to `<agent-runtime-dir>/logs/node-name`. Used when deploying multiple machine agents from a common directory. See Deploy Multiple Machine Agents From a Common Directory.

**System Property:** `-Dappdynamics.agent.runtime.dir`

**Environment Variable:** N/A

**Type:** String

**Default:** None
**Container Process Selector Blacklist Regex**

Any container with a process matching this regex would be ignored and not be registered in the Controller.

*System Property:* `-Dappdynamics.docker.container.process.selector.blacklist.regex`

*Environment Variable:* `APPDYNAMICS_DOCKER_CONTAINER_PROCESS_SELECTOR_BLACKLIST_REGEX`

<table>
<thead>
<tr>
<th>Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Required</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Controller Host**

This is the host name or the IP address of the AppDynamics Controller, e.g. 192.168.1.22 or myhost or myhost.abc.com. This is the same host that you use to access the AppDynamics browser-based user interface.

*Element in controller-info.xml:* `<controller-host>`

*System Property:* `-Dappdynamics.controller.hostName`

*Environment Variable:* `APPDYNAMICS_CONTROLLER_HOST_NAME`

<table>
<thead>
<tr>
<th>Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Required</td>
<td>No</td>
</tr>
</tbody>
</table>

**Controller Keystore Filename**

By default, the agent looks for a Java truststore file named cacerts.jks in the `conf` directory in the agent home. Use this property to enable full validation of Controller SSL certificates with a different Java truststore file. See Enable SSL for the Standalone Machine Agent.

*Element in controller-info.xml:* `<controller-keystore-filename>`

*System Property:* N/A

*Environment Variable:* N/A

<table>
<thead>
<tr>
<th>Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Required</td>
<td>No</td>
</tr>
</tbody>
</table>

**Controller Keystore Password**

The plain text or encrypted value of the Controller certificate password. To encrypt or obfuscate passwords, see Encrypt Agent Credentials.

*Element in controller-info.xml:* `<controller-keystore-password>`

*System Property:* N/A

*Environment Variable:* N/A

<table>
<thead>
<tr>
<th>Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Required</td>
<td>No</td>
</tr>
</tbody>
</table>
**Controller Port**
The HTTP(S) port of the AppDynamics Controller. This is the same port that you use to access the AppDynamics browser-based user interface. If the Controller SSL Enabled property is set to true, specify the HTTPS port of the Controller; otherwise, specify the HTTP port. See Controller SSL Enabled Property.

*Element in controller-info.xml:* `<controller-port>`

*System Property:* `-Dappdynamics.controller.port`

*Environment Variable:* `APPDYNAMICS_CONTROLLER_PORT`

*Type:* Positive Integer

*On-premises Default:* port 8090 for HTTP and port 8181 for HTTPS

*SaaS Default:* For the SaaS Controller Service, use port 443 for HTTPS connections.

*Required:* Yes, if the Enable Orchestration property is false.

If Enable Orchestration is true, and if the agent is deployed in a compute cloud instance created by an AppDynamics workflow, do not set the Controller port unless you want to override the auto-detected value. See Enable Orchestration Property.

**Controller SSL Enabled**
Specifies whether the agent should use SSL (HTTPS) to connect to the Controller. If SSL Enabled is true, set the Controller Port property to the HTTPS port of the Controller. See Controller Port Property.

*Element in controller-info.xml:* `<controller-ssl-enabled>`

*System Property:* `-Dappdynamics.controller.ssl.enabled`

*Environment Variable:* `APPDYNAMICS_CONTROLLER_SSL_ENABLED`

*Type:* Boolean

*Default:* false

*Required:* No

**Create Node if Absent**
Force the machine agent to create an APM node when the agent registers with the controller.

*Element in controller-info.xml:* `<create-node-if-absent>`

*System Property:* `-Dappdynamics.machine.agent.registration.createNodeIfAbsent`

*Environment Variable:* N/A

*Type:* Boolean

*Default:* true

*Required:* No. If you set the app/tier/node in your `controller-info.xml` file (existing upgrades or by accident), you can prevent the machine agent from creating APM nodes by setting this flag to false. See Standalone Machine Agent Installation Scenarios.

**Enable Docker Visibility**
To enable Docker Visibility on the agent, manually add the docker-enabled element in `controller-info.xml` setting, and set the flag to `true`.

*Element in controller-info.xml:* `<docker-enabled>true</docker-enabled>`

*System Property:* `-Dappdynamics.docker.enabled`

*Environment Variable:* `APPDYNAMICS_DOCKER_ENABLED`

*Type:* Boolean

*Default:* false

*Required:* Yes
Enable HTTP Listener

When set to true, this property enables the Machine Agent HTTP listener. You can send metrics to the Machine Agent using its HTTP listener. You can report metrics through the Machine Agent by making HTTP calls to the agent instead of piping to the agent through sysout.

Element in controller-info.xml: N/A
System Property: -Dmetric.http.listener
Environment Variable: N/A
Type: Boolean
Default: false
Required: No

Enable Orchestration

Enables the Machine Agent workflow task execution when set to True. It also enables auto-detection of the controller host and port when the app server is a compute cloud instance created by an AppDynamics orchestration workflow. In a cloud computing environment, auto-detection is necessary for the Create Machine tasks in the workflow to run correctly. The machine agent polls for task executions only when orchestration is enabled. If the host machine on which this agent resides is not created through AppDynamics workflow orchestration, this property should be set to false. Also, see Controller Host Property and Controller Port Property.

Element in controller-info.xml: <enable-orchestration>
System Property: N/A
Environment Variable: N/A
Type: Boolean
Default: false
Required: No

Force Default SSL Certificate Validation

Used to override the default behavior for SSL validation.

This property can have three states:

true: Forces the agent to perform full validation of the certificate sent by the controller, enabling the agent to enforce the SSL trust chain. Use this setting when a public certificate authority(CA) signs your Controller SSL certificate.

false: Forces the agent to perform minimal validation of the certificate. This property disables full validation of the Controller's SSL certificate. Use this setting when full validation of a SaaS certificate fails.

unspecified: The validation performed by the agent depends on the context:

- If the agent is connecting to a SaaS Controller, full validation is performed.
- If the agent is connecting to an on-premises Controller and the cacerts.jks file is present, then full validation is performed using the cacerts.jks file.
- If the agent is connecting to an on-premises Controller, and there is no cacerts.jks file, then minimal validation is performed

Element in controller-info.xml: N/A
System Property: -Dappdynamics.force.default.ssl.certificate.validation
Environment Variable: N/A
Type: Boolean
Default: None
Enable Dynamic Monitoring Mode (DMM)

When this option is enabled, the agent reports metrics based on the Dynamic Monitoring Mode specified for that agent in the Controller. When this option is disabled, the agent reports all metrics based on its local configuration; DMM settings on the Controller have no effect. Disabling DMM on an agent is recommended only for mission-critical servers and other machines for which you are sure you want to collect all available metrics at all times. For more information, see Dynamic Monitoring Mode and Server Visibility.

Element in controller-info.xml: <dynamic-monitoring-enabled>

System Property: appdynamics.machine.agent.dynamicMonitoring.enabled

Environment Variable: APPDYNAMICS_DYNAMIC_MONITORING_ENABLED

Type: Boolean

On-premises Default: True

SaaS Default: True

Required: No

HTTP Listener Port

To enable the Machine Agent HTTP listener, you must also specify the HTTP listener port.

Element in controller-info.xml: N/A

System Property: -Dmetric.http.listener.port

Environment Variable: N/A

Type: Numeric

Default: 8293

Required: Only if the HTTP listener is enabled.

Machine Hierarchy

You need a Server Visibility license to use this feature.

This setting enables you to group servers together into arbitrary hierarchies by specifying a hierarchical path to the server. The server hierarchy displays in the Metric Browser and on the Server Dashboard. The server hierarchy is also used to select subgroups of machines for health rules. The last element of the path indicates the server name, a name of your choice. This name appears as the Name on the Servers list. If the path contains spaces it must be enclosed in double-quotes. For more information, see Machine Agent Hierarchy.

Element in controller-info.xml: <machine-path>

System Property: -Dappdynamics.machine.agent.hierarchyPath

Environment Variable: APPDYNAMICS_MACHINE_HIERARCHY_PATH

Type: ASCII string with path elements that are separated by a "|" (bar).

Default: The value specified by Unique Host ID. If the last part of the machine hierarchy is empty, the Unique Host ID is the machine name. For example, if machine hierarchy is "Data Center 1|Rack 2|" and Unique host ID is "Host ID 3", then the machine hierarchy will become "Data Center 1|Rack 2|Host ID 3".

Required: No

Limitation: The length of the characters composing the machine-path up to, but not including, the last pipe cannot exceed 95 characters.

Examples:

- System Properties: -Dappdynamics.machine.agent.hierarchyPath= "Data Center 1|Rack 2|Machine3"
- controller-info.xml:
**Environment Variable:** APPDYNAMICS_MACHINE_HIERARCHY_PATH="Data Center 1|Rack 2|Machine3"

**Proxy Host**
The proxy host name or IP address. Proxy authentication cannot be used with SSL.

*Element in controller-info.xml:* N/A

*System Property:*
- `Dappdynamics.http.proxyHost`
- `Dappdynamics.https.proxyHost` *(Use if the agent is communicating with the Controller over SSL. Proxy authentication cannot be used with SSL.)*

*Environment Variable:* N/A

*Type:* String

*Default:* None

*Required:* Yes, if using a proxy to connect to the Controller. Otherwise, no.

**Proxy Password File**
The absolute path to the file containing the password of the user that is authenticated by the proxy host. The password must be the first line of the file and must be in clear (unencrypted) text. To encrypt or obfuscate passwords, see Encrypt Agent Credentials.

*Element in controller-info.xml:* N/A

*System Property:* `Dappdynamics.http.proxyPasswordFile`

*Environment Variable:* N/A

*Type:* String

*Default:* None

*Required:* No

*Example:* `-Dappdynamics.http.proxyPasswordFile=/path/to/file-with-password`

**Proxy Port**
The proxy HTTP(S) port. The default ports are 8090 (HTTP) and 443 (HTTPS).

*Element in controller-info.xml:* N/A

*System Property:*
- `Dappdynamics.http.proxyPort`
- `Dappdynamics.https.proxyPort` *(Use if the agent is communicating with the Controller over SSL. Proxy authentication cannot be used with SSL.)*

*Environment Variable:* N/A

*Type:* Positive Integer
Default: None
Required: Yes, if using a proxy to connect to the Controller. Otherwise, no.

**Proxy User Name**

The name of the user that is authenticated by the proxy host.

*Element in controller-info.xml:* N/A

*System Property:* `-Dappdynamics.http.proxyUser`

*Environment Variable:* N/A

*Type:* String

*Default:* None

*Required:* No

**Server Visibility Enabled**

Enable Server Visibility on the agent. This requires a Server Visibility license.

*Element in controller-info.xml:* `<sim-enabled>`

*System Property:* `-Dappdynamics.sim.enabled`

*Environment Variable:* `APPDYNAMICS_SIM_ENABLED`

*Type:* Boolean

*Default:* false

*Required:* Required to enable Server Visibility. See [Enable Server Visibility](#).

**Service Availability Update Interval**

This setting controls the time, in milliseconds, to wait between sending Service Availability periodic events to the controller. For support, licensing and other details, see [Service Availability](#).

*Element in controller-info.xml:* `<sam-event-update-interval-millis>`

*System Property:* `-Dappdynamics.machine.agent.sam.event.updateIntervalMillis`

*Environment Variable:* N/A

*Type:* Positive integer

*Default:* 300000 ms (5 minutes)

*Required:* No

**Unique Host ID**

This property logically partitions a single physical host or virtual machine. In the context of installing the machine agent, the unique Host ID property is not required. However, if you don’t define a unique Host ID, the Machine Agent uses the Java API to get the host ID. The results from the API can be inconsistent and in fact, the same JVM can sometimes return a different value for the same machine each time the machine agent is restarted. To avoid problems of this nature, we recommend that you set the value of unique Host ID to the host ID that you want to see in the UI.

*Element in controller-info.xml:* `<unique-host-id>`

*System Property:* `-Dappdynamics.agent.uniqueHostId`

*Environment Variable:* `APPDYNAMICS_AGENT_UNIQUE_HOST_ID`

*Type:* ASCII string without spaces and must be unique across the entire managed infrastructure.

*Default:* None
Required: Optional, but recommended.

**Use Simple Hostname**

By default (unless overridden with the uniqueHostId system property), the agent determines the host name of the OS it is running in by reverse DNS lookup. In some circumstances, this host name may be set as the fully qualified domain name of the host name. If this property is set to true, the agent removes any domain name and uses the simple hostname to identify the host. In cases where the host name is an IP address (which happens if the DNS lookup fails) the full IP address in string form is used. The host name is used in mapping metrics gathered by the machine agent to application nodes, see Unique Host ID Property.

*Element in controller-info.xml:* `<use-simple-hostname>`

*Type:* Boolean

*Default:* False

*Required:* No

*For example:* If this property is set to true `server.mydomain.com` becomes `server`.

**Independent Standalone Machine Agent Install Scenario**

Typically, you should only need to use the following properties if you are installing the Machine Agent on a server that does not have any AppDynamics app agents installed on it.

**Application Name**

The name of the logical business application that this JVM node belongs to. This is not the deployment name (ear/war/jar) on the application server. If a business application of the configured name does not exist, it is created automatically.

*Element in controller-info.xml:* `<application-name>`

*System Property:* `-Dappdynamics.agent.applicationName`

*Environment Variable:* `APPDYNAMICS_AGENT_APPLICATION_NAME`

*Type:* String

*Defaults:* None


**Node Name**

The name of the JVM node. When not specified, this defaults to Node1 for the Machine Agent.

*Element in controller-info.xml:* `<node-name>`

*System Property:* `-Dappdynamics.agent.nodeName`

*Environment Variable:* `APPDYNAMICS_AGENT_NODE_NAME`

*Type:* String

*Defaults:* None


**Tier Name**

The name of the logical tier that this JVM node belongs to. This is not the deployment name (ear/war/jar) on the application server. If a tier of the configured name does not exist, it is created automatically.

*Element in controller-info.xml:* `<tier-name>`

*System Property:* `-Dappdynamics.agent.tierName`

*Environment Variable:* `APPDYNAMICS_AGENT_TIER_NAME`
Type: String

Defaults: None

Controller Settings for Standalone Machine Agents

This topic describes Controller Admin settings that are specific to the Standalone Machine Agent. You need the root user password to change these settings.

You can change the amount of machine agent data you retain in the Controller database by changing the retention period for machine agent snapshots. Lowering the retention settings purges data that is aged out by the new retention setting and reduces the amount of data stored by the Controller on an ongoing basis. When tuning the size of the Controller database, this can be a good place to start.

**To Change the Controller Settings for Standalone Machine Agents**

1. Log in to the Controller administration console using the root user password. See Access the Administration Console.

   http://<controller host>:<port>/controller/admin.jsp

   Use the root user password to access the Admin console when the Controller is installed in single- or multi-tenant mode. For a discussion of the root password, see User Management.

2. Click Controller Settings.
3. Change the settings as needed and Save.
   Changes to the settings take effect the next time the agent is restarted and connects to the Controller.

### Controller Settings Reference for Standalone Machine Agent

<table>
<thead>
<tr>
<th>Property Name</th>
<th>About the property</th>
<th>Default</th>
<th>Allowed Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine.agent.in-progress.actions.timeout</td>
<td>This property determines how long the Controller waits for the agent to perform an IN-PROGRESS action before it stops the action.</td>
<td>86400000 ms (1 day)</td>
<td>1 to 840 hours (5 weeks)</td>
</tr>
<tr>
<td>machine.agent.max.new.actions.per.min</td>
<td>Maximum number of new actions dispatched per minute for each machine agent. This is the maximum number of &quot;runbook&quot; actions sent to the machine agent. For example, you may request the machine agent to run a script because of a health rule violation, if the number of requests in a one minute period is more than the preset value, the action is executed the next minute.</td>
<td>15 per minute</td>
<td>1 to 28999999</td>
</tr>
<tr>
<td>machine.agent.pending.actions.timeout</td>
<td>Determines how long the Controller waits for the agent to perform a PENDING action before it stops the action.</td>
<td>1800000 ms (12 hrs)</td>
<td></td>
</tr>
<tr>
<td>machine.agent.snapshots.buffer.size</td>
<td>Size in Megabytes of the in-memory buffer storing machine snapshots uploaded from agents prior to database flush</td>
<td>50 MB</td>
<td></td>
</tr>
<tr>
<td>machine.snapshots.retention.period</td>
<td>Time in hours to retain server snapshot data.</td>
<td>336 hours (2 weeks)</td>
<td>1 to 840 hours (5 weeks)</td>
</tr>
</tbody>
</table>
Enable SSL for Standalone Machine Agent

This topic describes how to configure the AppDynamics Standalone Machine Agent to connect to the Controller using SSL. It assumes that you use a SaaS Controller or have configured the on-premises Controller to use SSL.

The Standalone Machine Agent supports extending and enforcing the SSL trust chain when in SSL mode.

Plan SSL Configuration

Gather the following information:

- The Controller SSL port.
  - For SaaS Controllers the SSL port is 443.
  - For on-premises Controllers the default SSL port is 8181, but you may configure the Controller to listen for SSL on another port.
- The signature method for the Controller's SSL certificate:
  - A publicly known certificate authority (CA) signed the certificate. This applies for DigiCert, Verisign, Thawte, and other commercial CAs.
  - A CA internal to your organization signed the certificate. Some companies maintain internal certificate authorities to manage trust and encryption within their domain.
  - The Controller uses a self-signed certificate.

Establish Trust for the Controller's SSL Certificate

To establish trust between the Standalone Machine Agent and the AppDynamics Controller, you must create an agent truststore that contains the root certificate for the authority that signed the Controller's certificate.

1. Obtain one of the following root certificates:
   - DigiCert Global Root CA for the AppDynamics SaaS Controller
   - The root certificate for the publicly known certificate authority (CA) that signed the certificate for your on-premises Controller
   - The root certificate for the internal CA that signed the Controller certificate for your on-premises Controller
2. Run the Java keytool command to create the agent truststore:

   ```
   keytool -import -alias rootCA -file <root_certificate_file_name> -keystore cacerts.jks -storepass <truststore_password>
   ```

   For example:
keytool -import -alias rootCA -file
/usr/home/appdynamics/DigicertGlobalRootCA.pem -keystore
cacerts.jks -storepass MySecurePasssword

Note the truststore password; you will need this later to configure the Standalone Machine Agent.

3. Install the agent truststore to the agent configuration directory:

<machine_agent_home>/conf/

Secure the Standalone Machine Agent Truststore

AppDynamics recommends you take the following security measures to prevent tampering with the Standalone Machine Agent truststore:

- Secure the truststore file through filesystem permissions:
  - Make the agent truststore readable by any user.
  - Make the truststore owned by a privileged user.
  - Make the truststore writable only by the specified privileged user.
- Secure the controller-info configuration file so that it is only readable by the agent runtime user and only writable by a privileged user:
  - <machine_agent_home>/conf/controller-info.xml

Enable SSL for the Standalone Machine Agent


   - **Controller Host**: should be the same as either the Common Name or the Subject Alternative Name (SAN) in the certificate configured for the Controller.
     - <controller-host>common_name_in_certificate.com</controller-host>
   - **Controller Port**: the SSL port for the controller. 443 for AppDynamics SaaS.
     - <controller-port>443</controller-port>
   - **Controller SSL Enabled**: true.
     - <controller-ssl-enabled>true</controller-ssl-enabled>
   - **Controller SSL Password**: the plain text password for the agent truststore.
     - <controller-keystore-password>MySecurePassword</controller-keystore-password>

   If you have enabled the Secure Credential Store, encrypt the password you enter here. See Encrypt Agent Credentials.

- **Controller Keystore Filename**: path of the agent truststore relative to <machine_agent_home>/conf. Required if you use a truststore other than the default <machine_agent_home>/conf/cacerts.jks.
  - <controller-keystore-filename>../conf/cacerts.jks</controller-keystore-filename>

You can specify the Controller port and enable SSL for the Controller in the Standalone Machine Agent startup script, but you must specify the truststore password and filename in the controller-info.xml file.
2. Restart the Standalone Machine Agent.

**Sample controller-info.xml with SSL and Secure Credential Store encryption enabled**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<controller-info>
  <controller-host>mycompany.saas.appdynamics.com</controller-host>
  <controller-port>443</controller-port>
  <controller-ssl-enabled>true</controller-ssl-enabled>
    <!-- Encrypted Controller keystore / agent trust store password -->
    <controller-keystore-password>Tw49bd0hdCMBoQ5pfMMuYA/cA5B4pouVPkv48ovRm6c=</controller-keystore-password>
    <controller-keystore-filename>../../conf/cacerts.jks</controller-keystore-filename>
    ...
    <!-- Secure Credential Store configuration -->
    <!-- Enable the Secure Credential Store -->
    <use-encrypted-credentials>true</use-encrypted-credentials>
    <!-- Path to they secure credential keystore -->
    <credential-store-filename>/opt/appdynamics/secretKeyStore</credential-store-filename>
    <!-- Obfuscated secure credential keystore password -->
    <credential-store-password>n/8GvAZsKk4gM3Z6g+XQ1w==</credential-store-password>
</controller-info>
```

**Keystore Certificate Extractor Utility**

The Keystore Certificate Extractor Utility exports certificates from the Controller's Java keystore and writes them to an agent truststore. You can run this utility from the agent distribution on the Controller:

```
<controller_home>/appserver/glassfish/domains/domain1/appagent
```

1. Execute `kr.jar` and pass the following parameters:
   - The full path to the Controller's keystore:
1. The truststore output file name. By default, the Standalone Machine Agent looks for `cacerts.jks`.
2. The password for the Controller's certificate, which defaults to "changeit". If you don't include a password, the extractor applies the password "changeit" to the output truststore.

```
java -jar kr.jar
<controller_home>/appserver/glassfish/domains/domain1/config/keystore.jks cacerts.jks <controller_certificate_password>
```

2. Install the agent trust store to the agent configuration directory:

```
<machine_agent_home>/conf/
```
.NET Compatibility Mode

Note the following if you want to collect and view Machine or Server metrics on a server with Machine and .NET Agents installed:

- .NET Compatibility Mode is required for metrics from a Standalone Machine Agent and a .NET Agent on the same server to be visibly associated with the same node in the Controller UI.
- There might be differences in metric values reported by the Machine Agent and the .NET Agent due to different averaging rates and measurement methods.
- You must enable .NET Compatibility Mode on the Machine Agent.
- When .NET Compatibility Mode is enabled:
  - Events from Service Availability and Custom Extensions appear in the Servers tab but are not associated with .NET applications in the Controller UI.
  - The Controller UI might show the server name with “-java-MA” appended to the host ID. Thus, a Standalone Machine Agent with a <unique-host-id> set to “ABC” might appear as “ABC-java-MA” in the Controller UI. This is expected behavior; the “-java-MA” suffix indicates that the host has a Standalone Machine Agent running in .NET Compatibility Mode.
  - Do not specify the <application-name>, <tier-name>, or <node-name> properties on the Standalone Machine Agent. These properties are intended for use with a Standalone Machine Agent on a host that has no other AppDynamics agents installed.

To enable .NET Compatibility Mode:

1. You need to enable .NET Compatibility Mode on the agent. There are two methods to do this:
   - Set the following options in <machine_agent_home>/conf/controller-info.xml:
     a. `<dotnet-compatibility-mode>true</dotnet-compatibility-mode>`
        You might need to add this line to the controller-info.xml file.
     b. `<unique-host-id><unique-host-id-of-dot-net-agent></unique-host-id>`
        For both agents to report metrics to the same node, the Standalone Machine Agent must use the same case-sensitive Unique Host ID used by the .NET Agent. See FAQs and Troubleshooting.
   - Set the ./bin/machine-agent system property as follows:

2. Save the controller-info.xml file.
3. Start or restart the Machine Agent. You must do this to apply the previous changes.

You can also enable .NET Compatibility Mode on the Machine Agent by setting the ./.bin/machine-agent system property as follows:

```
./bin/machine-agent -Dappdynamics.machine.agent.dotnetCompatibilityMode=true
```
Using the Standalone Machine Agent, you can supplement the existing metrics in the AppDynamics Controller UI with your own custom metrics. There are many extensions currently available on the AppSphere Community site. Some are created by AppDynamics and some have been created by users.

Like built-in metrics, your custom metrics are subject to the following AppDynamics features:

- automatic baselines and anomaly detection
- availability for display on custom dashboards
- availability for use in policies
- visibility of all metrics in the Metric Browser and on the Infrastructure tab, where you can display external metrics along with AppDynamics metrics on the same graph

**Add New Custom Metrics**

To create custom metrics, you create a monitoring extension. In your extension, you define the name and path of your metric (where it appears in the metric browser tree), what type of metric it is (sum, average, and so on), and how the data for the metric should be rolled up as it ages. One agent can run many extensions, although you may need to increase the amount of memory for the agent JVM if you do. You can even have multiple copies of the same extension if they are in different directories.

A custom metric can be common across nodes or associated with a specific tier. When you create a metric, you specify the path in which it will appear in the metric tree. To make a common custom metric, use the root tree path Custom Metrics in your metric declaration. To make a tier-specific metric, specify the metric path associated with that component. For details, see the topics on creating Java or script-based custom metrics listed below.

If your application uses a large number of AppDynamics extensions with the Standalone Machine Agent, you might need to increase the size of the memory allocation as follows:

```
% <machine_agent_home>/bin/machine-agent -Xms64m
```
Types of Monitoring Extensions

You can implement custom metrics using the following mechanisms.

- **Using a script:**
  You can write a shell script (Linux and Unix-like systems) or batch file (Windows) to report custom metrics every minute to the Standalone Machine Agent. The Standalone Machine Agent passes these metrics on to the Controller.
  For more information, see Build a Monitoring Extension Using Scripts.

- **Using Java:**
  Your custom metrics might be too complicated to collect using a script. For example, you might need to perform complex calculations or call a third-party API to get the metrics. In this case, you can extend the JavaServersMonitor class to collect the metrics and report them to the Standalone Machine Agent. Your Java program extends the JavaServersMonitor class to provide your custom functionality.
  See Build a Monitoring Extension Using Java.

- **Using HTTP:**
  If you enable the agent HTTP listener, you can post HTTP requests to the Standalone Machine Agent to send it custom metrics every minute. This is done by starting the Standalone Machine Agent with a Jetty HTTP listener.
  See Standalone Machine Agent HTTP Listener for information on starting the HTTP listener and sending it metrics.
You can write a monitoring extension script (also known as a custom monitor or hardware 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 (Machine Agent). The Machine Agent passes these metrics to the Controller.

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

Review Existing Extensions

Before creating your own extension, review the extensions that have been created and shared among members of the AppDynamics community. New extensions are added continuously. It is possible that someone has already created exactly what you need or something close enough that you can download it and use it after making a few simple modifications.

The extensions are described and their source is available for free download at http://www.appdynamics.com/community/exchange/

Steps Overview

The general steps to create a monitoring extension using a script are the following:

1. Create your script. See Create the script file.
2. Create a monitor.xml configuration file. See Create the monitor.xml file.
3. Create a subdirectory, `<your_extension_dir>`, in `<machine_agent_home>/monitors`. See Create a directory under the Standalone Machine Agent monitors directory.
4. Copy your script file and the monitor.xml file into the new subdirectory.
5. Restart the Machine Agent.

Agent Configuration Requirements

Confirm that you have correctly configured the machine agent in the controller-info.xml file and on the agent start command on the command line. For information on configuring required and optional agent properties, see Standalone Machine Agent Configuration Properties.

Defining Your Metrics

Metric names must be unique within the same metric path but need not be unique for the entire metric hierarchy. It is a good idea to use short metric names so that the whole name is visible when displayed in the Metric Browser. Prepend the metric path to the metric name when you upload the metrics to the Controller.
Metric Processing Qualifiers

The Controller has various qualifiers for how it processes a metric with regard to aggregation, time rollup and tier rollup. There are three types of metric qualifiers:

1. Aggregation qualifier
2. Time roll-up qualifier
3. Cluster roll-up qualifier

In the script, specify the metric qualifiers after the name-value pair for the metric. A typical metric entry in the script file has the following structure:

```
name=<metric name>, value=<long value>, aggregator=<aggregator type>,
time-rollup=<time-rollup strategy>, cluster-rollup=<cluster-rollup strategy>
```

Aggregation Qualifier

The `aggregator` qualifier specifies how the Machine Agent aggregates the values reported during a one-minute period. Specify the aggregation qualifier as `aggregator="aggregator type"`. This value is an enumerated type. If no value is reported during that minute, no data is reported to the controller, and an UNCHANGED notice appears in the Machine Agent log for that metric. Valid values are:

<table>
<thead>
<tr>
<th>Aggregator Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVERAGE</td>
<td>Default. Average of all reported values in that minute.</td>
</tr>
<tr>
<td>SUM</td>
<td>Sum of all reported values in the minute, causes the metric to behave like a counter.</td>
</tr>
<tr>
<td>OBSERVATION</td>
<td>Last reported value in the minute.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Roll up Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVERAGE</td>
<td>Average of all one-minute values when adding it to the 10-minute granularity table; the average of all 10-minute values when adding it to the 60-minute granularity table.</td>
</tr>
<tr>
<td>SUM</td>
<td>Sum of all one-minute values when adding it to the 10-minute granularity table; the sum of all 10-minute values when adding it to the 60-minute granularity table.</td>
</tr>
<tr>
<td>CURRENT</td>
<td>Last reported one-minute value in that 10-minute interval; the last reported ten-minute value in that 60-minute interval.</td>
</tr>
</tbody>
</table>

Cluster Rollup 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:

<table>
<thead>
<tr>
<th>Roll up Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDIVIDUAL</td>
<td>Aggregates the metric value by averaging the metric values across each node in the tier.</td>
</tr>
<tr>
<td>COLLECTIVE</td>
<td>Aggregates the metric value by adding up the metric values for all the nodes in the tier.</td>
</tr>
</tbody>
</table>

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 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.
Add a Monitoring Extension Script

**Step 1: Create a subdirectory under the Standalone Machine Agent monitors directory**

The `<machine_agent_home>/monitors` directory is the repository for the Machine Agent extensions. For each new extension, create a subdirectory under the `/monitors` directory. The user running the agent requires read, write, and execute permissions to this subdirectory.

For example to create an extension that monitors open files in the JVM, create a subdirectory named "openfiles" under `<machine_agent_home>/monitors`. The structure looks like this:

```
MachineAgent directory
   ↓
Monitors subdirectory
   ↓
openfiles
   ↓
script file
   ↓
monitor.xml

CustomMonitor2
   ↓
script file
   ↓
monitor.xml

CustomMonitor3
   ↓
script file
   ↓
monitor.xml
```

**Step 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.

---

**For Windows custom metrics, PowerShell and VBScript are recommended over .bat files**

To generate custom metrics on Windows, it is good practice to use PowerShell and VBasic scripts instead of .bat files. When a standard Windows batch (.bat) script echoes metric names, it surrounds the names with quotes. The quotes will cause the Machine Agent to ignore these metrics. PowerShell and VBasic scripts do not have this issue.

---

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>, aggregator=<aggregator type>, time-rollup=<time-rollup strategy>,
   cluster-rollup=<cluster-rollup strategy>
   ```

   Use the following format:
2. Define the category of the metric, for example:
   a. Infrastructure (for the default hardware metrics, see Standalone Machine Agent)
   b. JVM
   c. Custom Metrics

Custom metrics must have the path prefixes:
   a. Custom Metrics
   b. Server|Component:<tier-name-or-tier-id>

Metrics with the Custom Metrics prefix are common across all tiers in your application. Metrics with the Server|Component:<tier-name-or-tier-id> prefix appear only under the specified tier.

To find the component ID of a tier, open the dashboard for the tier and inspect the URL. The ID appears as the component value in the URL, as shown:

```
Find the component number
```

The Machine Agent has to be associated with the target/destination for the metrics. If you try to publish metrics to a Tier that is not associated with the Machine Agent, the metrics can not be reported.

The "|" character separates the branches in the metric hierarchy, telling the Controller where the metric should appear in the metric tree:

```
Custom Metrics|Hardware Resources|Disks|Total Disk Usage %
Custom Metrics|Hardware Resources|Disks|Disk 1|Current Disk Usage %
```

You can insert a custom metric alongside an existing type of metric. For example, the following declaration causes the custom metric named pool usage to appear alongside the JMX metrics:

- Server|Component:18|JMX|Pool|First|pool usage

The metric can then be used in health rules as would other types of JMX metrics.

To monitor multiple metrics with the same script file, have the script write a different line for each one to STDOUT, such as the following:
Step 3: Copy the script file to the subdirectory 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.

Step 4: Create the monitor.xml file

For each custom monitoring extension script 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.

```xml
<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>continuous</execution-style>
    <name>Run</name>
    <type>executable</type>
    <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.

1. Select the execution style from one of the following:
<table>
<thead>
<tr>
<th>Execution Style</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>continuous</td>
<td>Choose if you want data collection averaged over time – for example, average CPU usage over a minute. For the monitor to be declared as 'continuous', the script should also run in an infinite loop. This ensures that the script keeps running until the Standalone Machine Agent process is terminated.</td>
<td>while [ 1 ]; do ... the actual script goes here ... sleep 60 done</td>
</tr>
</tbody>
</table>
| periodic        | Choose to report data from system performance counters periodically. The periodic task runs every minute by default and the data is aggregated. To specify a different frequency, use the execution-frequency-in-seconds element. The execution frequency must be less than 60. For periodic execution style, you can also specify the timeout setting as shown in the example. | <monitor-run-task>
  ...<execution-style>periodic</execution-style>
  <execution-frequency-in-seconds>30</execution-frequency-in-seconds>
  <execution-timeout-in-secs>30</execution-timeout-in-secs>
  ...
</monitor-run-task> |

2. Add the name of your 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").`

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

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

**Step 5: Copy the monitor.xml file to the subdirectory created in Step 1**

**Step 6: Restart the Standalone Machine Agent**

**Required Agent Properties**
Ensure that you have correctly configured the agent in the controller-info.xml file and on the agent start command on the command line. For information on configuring required and optional agent properties, see Database Agent Configuration Properties.

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.
```

**Step 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. Here are two examples:

   Modify this UNIX 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
   ```

   The following Windows .bat example reports a metric to the Controller if it a Java process is running on the machine. **NOTE:** To generate custom metrics on Windows, it is good practice to use PowerShell or VBasic scripts rather than .bat scripts. When a standard Windows .bat script echoes metric names, it surrounds the names with quotes. The quotes will cause the Machine Agent to ignore these metrics. PowerShell and VBasic scripts do not have this issue.

   ```
   SETLOCAL enabledelayedexpansion
   
   REM Check to see if there is a java process running
   TASKLIST /FI "IMAGENAME eq java.exe" 2>NUL | find /I /N "java.exe">NUL
   if "%ERRORLEVEL%"=="0" (  
       SET metric="Custom Metrics|Process|java|Running,value=1"
       REM This strips the quotes that are added by the bat script
       REM so that the machine agent can understand the metric
       @echo !metric:"=!
   )
   ```

3. Create the following `monitor.xml` file and point it to the UNIX script shown 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>
You can create Java monitoring extensions that enable the Standalone Machine Agent and Server Visibility to collect custom metrics, which you define and provide, and report them to the Controller. This is an alternative to adding monitoring extensions using scripts.

When you capture custom metrics with a monitoring extension, they are supported by the same AppDynamics services that you get for the standard metrics captured with the AppDynamics application and machine agents. These services include automatic baselining, anomaly detection, display in the Metric Browser, availability for display on custom dashboards and availability for use in policies to trigger alerts and other actions.

This topic describes how to create a monitoring extension in Java.

To the agent, a monitoring extension is a task that runs on a fixed schedule and collects metrics.

### Before You Begin

Before creating your own extension from scratch, review the extensions on the AppDynamics community. The extensions are described and their source is available for free download at:

https://github.com/Appdynamics/

New extensions are constantly being added. It is possible that someone has already created what you need or something close enough that you can download it and use it after making a few simple modifications.

### Overview of Creating a Monitoring Extension

To create a monitoring extension in Java:

1. Create your extension class. See Create the Monitoring Extension.
2. Create a monitor.xml configuration file. See Create the monitor.xml File.
3. Create a subdirectory `<your_extension_dir>` under `<agent_home>/monitors`.
4. Place the extension class file and the monitor.xml file (plus any dependent jar files) in `<your_extension_dir>`.
5. Enter the Controller access information and credentials. See Configure the Standalone Machine Agent.

   Make sure that your controller-info and command line parameters are correctly configured. Required properties include Controller name, port number, and account access key.


### Create the Monitoring Extension Class

Create a monitoring extension class by extending the AManagedMonitor class in the com.singularity.ee.agent.systemagent.api package. This package is included in the MachineAgent.jar file.
Your monitor extension class performs these tasks:

- Populates a hash map with the values of the metrics that you want to add to AppDynamics. How you obtain these metrics is specific to your environment and to the source from which you derive your custom metrics.
- Defines the type of metrics to collect using the MetricWriter class. See Metric Processing Qualifier.
- Uploads the metrics to the Controller using the `execute()` method of the AManagedMonitor class.

Example source code is here: Extension_Class_Source.txt.

**Metric Path**

All custom metrics processed by the Machine Agent and Server Visibility appear in the Metric Browser > Application Infrastructure Performance. Use the "|" character to specify the path from Application Infrastructure Performance to the custom metric. If the metrics apply to a specific tier, use the metric path for the tier, with "Component" followed by a colon ":" and the tier name or tier ID.

For example, suppose you want to associate a metric with tier AccountService. In this case, you would specify the metric path as follows:

- `Server|Component:AccountService|Custom Metrics|Path`

The metric would then appear under the following tree in the Metric Browser:

![Metric Tree Diagram]

You can report a custom metric only to the tier that is associated with the Standalone Machine Agent. If you try to publish metrics to a different tier, the metrics will not be reported.

You can insert a custom metric alongside an existing type of metric. For example, the following declaration causes the custom metric named pool usage to appear alongside the JMX metrics:

- `name="Server|Component:<tier-name>|JMX|Pool|First|pool usage",value=10`

The metric can then be used in health rules as would other types of JMX metrics.

You can test the appearance of your custom metric in the Controller API by posting the metric data to the Standalone Machine Agent's REST API. Pass the path, name type and values of the metric as URL arguments. See Standalone Machine Agent HTTP Listener for more information.

**Metric Names**

Metric names must be unique within the same metric path but need not be unique for the entire metric hierarchy.

Try to use short metric names so that they are visible when they are displayed in the Metric Browser.

Prepend the metric path to the metric name when you upload the metrics to the Controller.

**Metric Processing Qualifier**
The Controller has various qualifiers for how it processes a metric with regard to aggregation, time rollup and tier rollup.

There are three types of metric qualifiers:

- Aggregation qualifier
- Time roll-up qualifier
- Cluster roll-up qualifier

You specify these options with the enumerated types provided by the MetricWriter class. These types are defined below.

**Aggregation Qualifier**

The aggregator qualifier specifies how the Machine Agent aggregates the values reported during a one-minute period.

Specify the aggregation qualifier as `aggregator="aggregator type"`

This value is an enumerated type. Valid values are:

<table>
<thead>
<tr>
<th>Aggregator Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>METRIC_AGGREGATION_TYPE_AVERAGE</td>
<td>Default. Average of all reported values in that minute.</td>
</tr>
<tr>
<td>METRIC_AGGREGATION_TYPE_SUM</td>
<td>Sum of all reported values in that minute, causes the metric to behave like a counter.</td>
</tr>
<tr>
<td>METRIC_AGGREGATION_TYPE_OBSERVATION</td>
<td>Last reported value in the minute. If no value is reported in that minute, the last reported value is used.</td>
</tr>
</tbody>
</table>

**Time Roll Up**

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.

<table>
<thead>
<tr>
<th>Roll up Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>METRIC_TIME_ROLLUP_TYPE_AVERAGE</td>
<td>Average of all one-minute data points when adding it to the 10-minute or 60-minute granularity table.</td>
</tr>
<tr>
<td>METRIC_TIME_ROLLUP_TYPE_SUM</td>
<td>Sum of all one-minute data points when adding it to the 10-minute or 60-minute granularity table.</td>
</tr>
<tr>
<td>METRIC_TIME_ROLLUP_TYPE_CURRENT</td>
<td>Last reported one-minute data point in that 10-minute or 60-minute interval.</td>
</tr>
</tbody>
</table>

**Cluster Roll Up**

The cluster-rollup qualifier specifies how the controller aggregates metric values in a tier.

<table>
<thead>
<tr>
<th>Roll up Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>METRIC_CLUSTER_ROLLUP_TYPE_INDIVIDUAL</td>
<td>Aggregates the metric value by averaging the metric values across each node in the tier.</td>
</tr>
<tr>
<td>METRIC_CLUSTER_ROLLUP_TYPE_COLLECTIVE</td>
<td>Aggregates the metric value by adding up the metric values for all the nodes in the tier.</td>
</tr>
</tbody>
</table>

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 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.

**Sample Monitoring Extension Class**

The NGinXMonitor class gets the following metrics from the Nginx Web Server and adds them to the metrics reported by AppDynamics:

- Active Connections: number of active connections
- Accepts: number of accepted requests
- Handled: number of handled requests
Create monitor.xml

Create a monitor.xml file with a <monitor> element to configure how the Machine Agent executes the extension.

1. Set the <name> to the name of your Java monitoring extension class.
2. Set the <type> to managed.
3. The <execution-style> can be continuous or periodic.
   - Continuous means to collect the metrics averaged over time; for example, average CPU usage per minute. In continuous execution, the Machine Agent invokes the extension once and the program runs continuously, returning data every 60 seconds.
   - Periodic means to invoke the monitor at a specified frequency. In periodic execution, the Machine Agent invokes the extension, runs it briefly, and returns the data on the schedule set by the <execution-frequency-in-seconds> element.

   Do not set the <execution-frequency-in-seconds> higher than 300 seconds (five minutes). The extension must collect metrics at least once every five minutes.

4. If you chose periodic for the execution style, set the frequency of collection in <execution-timeout-in-secs> element. The default frequency is 60 seconds. If you chose continuous, this setting is ignored.
5. Set the <type> in the <monitor-run-task> child element to java.
6. Set the <execution-timeout-in-secs> to the number of seconds before the extension times out.
7. Specify any required task arguments in the <task-arguments> element. The default arguments that are specified here are the only arguments that the extension uses. They are not set anywhere else.
8. Set the <classpath> to the jar file that contains your extension's classes. Include any dependent jar files, separated by semicolons.
9. Set the <impl-class> to the full path of the class that the Machine Agent invokes.

Sample monitor.xml Files

This attached monitor.xml file configures the NGinXMonitor monitoring extension. This extension executes every 60 seconds.

This attached monitor.xml file configures the MysqlMonitor. This monitor executes every 60 seconds, has four required task arguments and one optional task argument and one dependent jar file.
Standalone Machine Agent HTTP Listener

You can send metrics to the Standalone Machine Agent using its HTTP listener. You can report metrics through the Standalone Machine Agent by making HTTP calls to the agent instead of piping to the agent through sysout.

Activate the HTTP Listener

The HTTP listener is not enabled by default. To activate the HTTP listener, restart the Standalone Machine Agent and set the metric.http.listener system property to true. Optionally, you can specify the host and port for the listener with system properties.

The -D system properties are as follows:

- metric.http.listener: Required. Set to true.
- metric.http.listener.port: Optional. Set to the port to be used, defaults to 8293.
- metric.http.listener.host: Optional. This describes which interface to accept requests on. You can set it as follows:
  - If you do not specify the metric.http.listener.host, it defaults to localhost, which means requests are only accepted from localhost.
  - Set to 0.0.0.0 to accept on all interfaces (including from remote clients)
  - Set to the IP address of the specific interface to accept requests on.

```
<machine_agent_home>/bin/machine-agent -Dmetric.http.listener=true
-Dmetric.http.listener.port=<port_number>
-Dmetric.http.listener.host=0.0.0.0
```

If starting the agent by invoking the Machine Agent JAR, be sure to put the options before the JAR name in your start up command. For example:

```
java -Dmetric.http.listener=true -jar
MACHINE_AGENT_HOME/machineagent.jar
```

Create Metrics

Use to post custom metrics to the Machine Agent for uploading to the Controller. Define one or more metrics in the body of the request as JSON data.

**URI**

POST /api/v1/metrics

**Metric Definition Fields**
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metricName</td>
<td>Name for the metric as it will appear in the Controller UI.</td>
</tr>
<tr>
<td>aggregatorType</td>
<td>How the metrics should be aggregated. Options are:</td>
</tr>
<tr>
<td></td>
<td>• AVERAGE: The average of all one-minute data points when adding it to the 10-minute or 60-minute granularity table.</td>
</tr>
<tr>
<td></td>
<td>• SUM: The sum of all one-minute data points when adding it to the 10-minute or 60-minute granularity table.</td>
</tr>
<tr>
<td></td>
<td>• OBSERVATION: Last reported one-minute data point in that 10-minute or 60-minute interval.</td>
</tr>
<tr>
<td>Value</td>
<td>A 64-bit integer value for the metric.</td>
</tr>
</tbody>
</table>

**Format**

```json
POST /api/v1/metrics
[

{

    "metricName": "Custom Metrics|Test|MetricFromRESTClient1",

    "aggregatorType": "AVERAGE",

    "value": 10

}

]
```

**Create Events**

Use to post custom events to the Machine Agent for uploading to the Controller. Define one or more events in the body of the request as JSON data.

**URI**

POST /api/v1/events

**Event Definition Fields**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventSeverity</td>
<td>Severity of the event, from INFO, WARN, ERROR.</td>
</tr>
<tr>
<td>type</td>
<td>A string describing the event subtype. The event will be of type CUSTOM, subtype will be the value of this parameter.</td>
</tr>
<tr>
<td>summaryMessage</td>
<td>A summary of the event.</td>
</tr>
</tbody>
</table>
**properties**

Event properties. These properties are retrieved along with events by the Controller in a given query and provide a means for filtering the events. The maximum size of a key is 500 chars and the maximum size of a value is 5000 characters. The values can be a numeric or string value.

**details**

Arbitrary key-value details for the event; similar in constraints to properties but are retrieved in a separate call. Use this to store details that should only be retrieved when requested by the user, which avoids the expense of retrieving this data in the usual event calls.

---

**Format**

```
POST /api/v1/events
[
    {
        "eventSeverity": <event_severity>,
        "type": "<event_type>",
        "summaryMessage": "<event_summary>",
        "properties": {
            "<key>": {
                <user-specified_object>
            },
            ...
        },
        "details": {
            "<key>": "<value>
        }
    },
    {
        "eventSeverity": <event_severity>,
        "type": "<event_type>",
        "summaryMessage": "<event_summary>",
        "properties": {
            "<key>": {
                <user-specified_object>
            },
            ...
        },
        "details": {
            "<key>": "<value>
        }
    },...
]
```

---

**Legacy Machine Agent HTTP APIs**

The following API endpoints are supported for backward compatibility, but will not be extended or enhanced in future versions.

**Upload Metrics**

You can use GET or POST to upload metrics to the Metric Browser under Application Performance > Tier where the tier is the one
defined for the Standalone Machine Agent.

The format for GET is:

```
GET /machineagent/metrics
```

For example:

```
http://host:port/machineagent/metrics?name=Custom Metrics|Test|My Metric&value=42&type=average
```

The format for POST is:

```
POST /machineagent/metrics
```

with header:

```
Content-Type: application/xml
```

with body content:

```
<?xml version="1.0"?><request>
  <metric name="[name of metric 1]", type="[aggregation type]",
   value="[value of metric 1]" />
  <metric name="[name of metric 2]", type="[aggregation type]",
   value="[value of metric 2]" />
  ...
  <metric name="[name of metric n]", type="[aggregation type]",
   value="[value of metric n]" />
</request>
```

Example:
http://host:port/machineagent/metrics

Example of body content:

```xml
<request>
  <metric name="Custom Metrics|Test|My Metric 1", type="AVERAGE", value="22" />
  <metric name="Custom Metrics|Test|My Metric 2", type="SUM", value="98737" />
  <metric name="Custom Metrics|Test|My Metric 3", type="CURRENT", value="93" />
</request>
```

Valid values for type are:

- 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.

**Upload events**

Send events using HTTP GET requests to upload events to the Machine Agent. The format is:

```
GET /machineagent/event
```

For example:

```
http://localhost:8293/machineagent/event?type=<event_type>&summary=<summary_text>
```

Event_type is one of the following:

- error
- info
- warning
Administer the Standalone Machine Agent

Related pages:
- Standalone Machine Agent
- Standalone Machine Agent Requirements and Supported Environments
- Install the Standalone Machine Agent

The following topics describe procedures and provide reference material relevant to the AppDynamics Pro Administrator.

- Manage Machine Agents
- Upgrade the Standalone Machine Agent
- Uninstall the Standalone Machine Agent
- FAQs and Troubleshooting
Manage Machine Agents

On this page:

- View Details
- Available Actions
  - Reset Selected Machine Agent
  - View Machine
  - Disable and Enable Agents
  - Delete Selected Machine Agent

Related pages:
- Manage App Agents

View Details

Using an AppDynamics Administrator account, click Settings > AppDynamics Agents > Machine Agents to see details and manage your installed Machine Agents.

- Hostname: The name of the machine on which the agent is installed.
- Version: The version of the installed Machine Agent software.
- Applications: The name of the application with which the agent is associated.
- Enabled: Whether the agent is enabled.

Available Actions

Reset Selected Machine Agent

Restarts the selected Machine Agent and runs it with any updated configuration files on the agent host and any changes made on the Controller that affect the agent. The reset functionality works only for pre 4.2 Machine Agents.

View Machine

Shows summary details about the Machine Agent as follows:

- Name: The name of the machine on which the agent is installed.
- IP Address: The IP address of the machine on which the agent is installed.
- OS: The operating system running the agent.
- Status: This field displays the following information: Initialized
- Install Time: Date and time agent was installed.
- Last Start Time: Date and time agent was last restarted.
**Disable and Enable Agents**

Disabling the agent temporarily suspends its collection of metrics. The agent can be quickly re-enabled when you want.

**Permission:**
You must have the application-level permission, **configure agent properties** to enable and disable agents.

When the Standalone Machine Agent is disabled, the agent stops reporting hardware metrics and metrics from any extensions installed on the agent until you re-enable it. The disabled Standalone Machine Agent no longer reports events and any installed extensions are also stopped. While disabled, the agent continues to consume a license. The agent maintains a "heartbeat" connection to the Controller so that it can be 'enabled' again quickly. The agent persists its enabled/disabled state even after the Standalone Machine Agent is restarted. Once re-enabled, all normal Standalone Machine Agent activities resume.

**Delete Selected Machine Agent**

The purpose of the delete functionality on the admin page is to delete the underlying machine metadata for an orphaned machine. A machine is considered "orphaned" when it is no longer associated with any APM nodes or with Server Availability. You can not delete a machine that is not an orphan. To delete a machine that is not an orphan, you must first delete all nodes on that machine and delete the machine from the Servers list. You can select multiple machines for the delete action.

You can also delete a machine agent from the **Servers List**. Note the following:

- Server Visibility Administrator permission is required to delete machines.
- You can delete machines from the Servers list regardless of whether they have Server Visibility enabled. You can select multiple machines for the delete action.
- Deleting a machine from the Servers 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.
Upgrade the Standalone Machine Agent

On this page:
- Perform the Upgrade

Related pages:
- Standalone Machine Agent Configuration Properties
- Agent and Controller Compatibility
- Upgrade the Controller Using the Enterprise Console

This topic describes upgrading an existing installation of the Machine Agent. The instructions have been tested and certified to work for Machine Agent versions 4.3 and above.

For a new installation, see Install the Standalone Machine Agent.

4.3 or later version of Machine Agent Recommended
AppDynamics recommends using version 4.3 or later, of the Machine Agent for this release. Version 4.1, or earlier, is no longer supported. Although Machine Agents may be compatible with future releases of the Controller, AppDynamics only provides support for Machine Agent versions with the Controller for two years after the release of the agent.

Perform the Upgrade

1. Do the following (as needed) before you update the agent:
   a. Review the Release Notes for changes that affect your environment.
   b. Back up the `<machine_agent_home>` directory to a `<machine_agent_home_BACKUP>` so you can revert to the previous installation if required.
   c. If you are upgrading the Controller as well as the agents: Upgrade the Controller Using the Enterprise Console and then return to this procedure.
   d. If you have been collecting Log Analytics data by running the Analytics Agent via the Machine Agent: Do the following.
      i. Preserve the old watermark file so that you can copy it to the new `<analytics-agent-home>`.
         Machine Agent location: `<machine-agent-home>/monitors/analytics-agent/conf/watermark`
         Analytics Agent location: `<analytics-agent-home>/conf/watermark`
      ii. Preserve your pre-existing job files so that you can copy them to the new `<analytics-agent-home>`.
         Analytics Agent location: `<analytics-agent-home>/conf/job`

2. Download the installer suitable for your OS environment from the AppDynamics Download Center (https://appdynamics.com/download).

3. Stop all instances of the Standalone Machine Agent that are currently running from the `<machine_agent_home>` that you want to upgrade. See Start and Stop the Standalone Machine Agent.

4. Update your Machine Agent installation as follows:
   - RPM-based Linux installation:
     Update the existing package. For example:
     ```bash
     sudo rpm -U <package>
     ```
   - RPM-based installation on a systemd Linux system (such as CentOS 7, Ubuntu, newer Fedora Core systems, newer OpenSUSE systems, and newer SUSE Enterprise Linux systems):
     If you don’t know if your Linux installation is using systemd, do a clean install to avoid issues: remove the old package and then install the new one.
   - For all other scenarios:
     First, delete the `<machine_agent_home>` for the old agent. Make sure you retain the `<machine_agent_home_BACKUP>` you created in step 1.
     Then install the Machine Agent as described Linux Install Using the RPM Package for your specific installation in Install the Standalone Machine Agent.

     If you are running the Analytics Agent as an extension to the Machine Agent, failure to clear out the old directory or use a new location may cause your embedded Analytics Agent to fail to start.

5. Do the following (as needed) before you start the updated agent.
   a. If you are updating on Windows: When you install the new Machine Agent in the existing directory, you may see two "InstallService" and "UninstallService" files. Use the .vbs files (VbScript) and delete the .cmd files (Windows Command Script), which are leftover from your previous version.
   b. If the agent install package includes a new JRE: Update any related custom scripts that use the Machine Agent JRE.
   c. Update all users who have the Account Administrator role for the account add the appropriate SIM roles (SIM admin and SIM user) to other users in the account as needed.

6. Do the following (as needed) before you start the updated agent.
   a. If you are updating on Windows: When you install the new Machine Agent in the existing directory, you may see two "InstallService" and "UninstallService" files. Use the .vbs files (VbScript) and delete the .cmd files (Windows Command Script), which are leftover from your previous version.
   b. If the agent install package includes a new JRE: Update any related custom scripts that use the Machine Agent JRE.
   c. Update all users who have the Account Administrator role for the account add the appropriate SIM roles (SIM admin and SIM user) to other users in the account as needed.

7. Start the agent, verify the agent installation, and verify that the agent is reporting to the Controller as described for your specific installation in Install the Standalone Machine Agent.
Uninstall the Standalone Machine Agent

On this page:
- Remove the Agent Service
- Deleting Using AppDynamics Manage Agents

To uninstall the Standalone Machine Agent, use these steps:

1. Stop the machine agent (or service) For the commands for your environment, see Start and Stop the Standalone Machine Agent.
2. If you installed the Machine Agent as a service, delete the service. See commands below. For Windows and Mac OS X, the commands to remove the service also stop the service cleanly.
3. Delete the <machine-agent-install> installation directory.

Remove the Agent Service

**Linux RPM**

Execute the following:

```
sudo rpm -ev appdynamics-machine-agent
```

**Windows**

Execute the following:

```
> UninstallService.vbs
```

**Mac OS X**

Execute the following:

```
sudo launchctl unload -w
<machine_agent_home>/com.appdynamics.machineagent.plist
```

Deleting Using AppDynamics Manage Agents

If you delete a Machine Agent from the Controller UI, as described in Manage Machine Agents, but do not shut down the JVM that the Machine Agent runs on, the Machine Agent reappears in the UI the next time it connects to the Controller. To prevent a Machine Agent from connecting to the Controller, you need to remove the Machine Agent settings from the JVM configuration. This frees the
license associated with the agent in the Controller and makes it available for use by another Machine Agent.
FAQs and Troubleshooting

On this page:
- Machine Agent and App Agent on the Same Machine, but Report to Different Nodes
- Identify the Standalone Machine Agent Process
- Metric Values for Disk and Network are Zero
- Flow Maps and the Standalone Machine Agent
- Server Health Indicator

Machine Agent and App Agent on the Same Machine, but Report to Different Nodes

Issue

A machine has a Standalone Machine Agent and an App Server Agent running together, but the agents report to different nodes. When you look in the Metric Browser, for example, application metrics appear under node abc but machine metrics appear under node abc.mydomain.com.

Resolution

First, check to see if the hostnames match. If they don’t:

1. Set the `<unique-host-id>` for the Machine Agent to the same `<unique-host-id>` as the App Server Agent.
2. Restart the Machine Agent to apply the previous changes.

The `<unique-host-id>` on the Machine Agent and the `<unique-host-id>` for the App Server Agent must match exactly. These settings are case-sensitive and ensure that both agents report metrics to the same node.

To verify that the hostnames match:

1. Click the gear icon (⚙️) in the Controller menu and choose AppDynamics Agents.
2. Select the Machine Agents tab.
3. Check the Applications field for the agent name.

   In addition, for consistency in naming in the Servers tab, use the `<machine path>` attribute to set the hostname. By using the machine path, you don’t have to use the fully qualified domain name (FQDN) for some agents, and the hostname for others. The reported hostname is still available to view by checking the option in the Agents Admin view.

   ![AppDynamics Agents](image)

4. Check the ID for the App Server Agent:
   a. Go to the App Server Agents tab.
   b. Search for the App Server Agent to which you want to associate the Machine Agent.
   c. Check the Unique Host ID field for the App Server Agent installed on that machine.
5. Set the `<unique-host-id>` for the Machine Agent to the same `<unique-host-id>` as the App Server Agent.
6. Restart the Machine Agent to apply the previous changes. Your Agents will appear as associated after a few minutes.

Identify the Standalone Machine Agent Process

Linux and Unix-like systems
To identify which process is running the machine agent, use the following command:

```
ps -ef | grep machine
```

**Windows systems**

In the Windows Services Application, look for the AppDynamics Machine Agent service.

**Metric Values for Disk and Network are Zero**

When a 32-bit JRE is used with a 64-bit operating system, you may see values of zero for disk and network metrics. To solve this problem, use a 64-bit JRE with the 64-bit operating system.

**Flow Maps and the Standalone Machine Agent**

The Machine Agent monitors a particular machine (also called "host server") and not a particular application server. The machine agent can, therefore, report to multiple nodes running on the same machine. A flow map, on the other hand, displays the communication between different nodes during application execution, or the business transaction flow from tier to tier. A Machine Agent cannot be a part of the flow and therefore is not shown in the flow map.

**Server Health Indicator**

Health rule violations based on metrics monitored by the machine agent are included in the health indicator on the various application, tier, and node dashboards. The health indicator is driven by health rule violations in the given time period configured on hardware metrics collected by the Machine Agent. Some health rules are configured by default. To configure additional health rules, see **Configure Health Rules**. For more details on Server Availability and health rules, see **Configure Health Rules to Monitor Servers**.
Server Visibility

The AppDynamics Server Visibility module uses the Machine Agent to provide extended hardware metrics and Service Availability Monitoring. Server Visibility 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.

Before you Start

Server Visibility requires the following:

- The monitored machine must have a Server Visibility license.
- Server Visibility must be enabled on the Machine Agent. See Enable Server Visibility.

To view basic hardware metrics without Server Visibility, see Viewing Hardware Metrics.

Server Visibility is currently available for Linux, Windows, and Solaris. You must do the following before you enable Server Visibility:

1. Review the Server Visibility Requirements and Supported Environments.
2. Install and configure the Machine Agent on a machine with a supported OS as described in Install the Standalone Machine Agent.

Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Server Visibility</th>
<th>Machine Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Platforms</td>
<td>Linux, Windows, Solaris</td>
<td>Linux and Unix-like systems, MacOS, and Windows</td>
</tr>
<tr>
<td>Extensions and Custom Metrics</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Monitor Docker container metrics from the Docker host</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Tier Metric Correlator</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Dynamic Monitoring Mode</td>
<td>Yes</td>
<td>No</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 Visibility</td>
<td>Machine Agent</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Yes</td>
<td>Functionality is limited to viewing Machine Agent metrics in the Servers List</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----</td>
<td>--------------------------------------------------------------------------------</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>

**Permissions**

Server Visibility provides two predefined roles and two related permissions:

- Server Visibility Administrator
- Server Visibility User
- **View Server Visibility** permission - enables configuring Service Availability
- **Configure Server Visibility** permission - enables configuring Service Availability

For more details, see Server Visibility Permissions.

**Server Visibility 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.

The **Servers > Containers** window lists the monitored containers used by an application.

**Using the Server Visibility UI**

The Server Visibility 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.
Click a window name to display that window

Select the column you want to display

Click a column heading to sort the table by that column
Server Visibility Requirements and Supported Environments

On this page:
- System Requirements
- Limitations
- Server Visibility on Windows

Related pages:
- Standalone Machine Agent Requirements and Supported Environments
- Enable Server Visibility
- .NET Compatibility Mode

Please review the following guidelines and notes if you want to enable Server Visibility.

System Requirements

Server Visibility requires a supported version of the Standalone Machine Agent to be installed on the monitored server. See Standalone Machine Agent Requirements and Supported Environments.

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

You can install and run Standalone Machine Agents on other supported platforms, but Server Visibility features are not available.

Limitations

- Total number of machine agents and containers reporting to a single Controller is 30,000.
- Total number of app agents reporting to a single Controller is 100,000.
- Monitored servers are considered stale 30 days after they go offline. They are purged from the Controller database when that time limit is reached.
- Docker Visibility is supported on Linux only.

Server Visibility on Windows

Note the following.

- It is good practice to check the server frequently to make sure it has the latest Windows updates installed. WMI (Windows Management Instrumentation) updates are especially important because the agent uses WMI to collect Server Visibility metrics.
- The monitored server should have at least 4 cores or multiple CPUs. WMI processes can be highly resource-intensive, especially on Windows Server 2012.
- If you do not have a Server Visibility license, or if you want to collect Basic metrics only, configure the agent to use the JavaHardwareMonitor extension.
- If you do not have a Server Visibility license, or you want to collect Basic metrics only, should not be enabled on the agent.
- If you want to enable Server Visibility on a Windows server with a .NET APM agent installed, you must enable .NET Compatibility Mode on both the Controller and the Standalone Machine Agent.
Enable Server Visibility

You must do the following before you enable Server Visibility:

1. Review the Server Visibility Requirements and Supported Environments.
2. Install and configure the Standalone Machine Agent on a machine with a supported OS as described in Install the Standalone Machine Agent.

To enable Server Visibility

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 Visibility

This section describes the Server Visibility 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 Process Metrics
- Server Volumes Metrics
- Server Network Metrics
- Navigating Between Server and Application Contexts
Discover Normal Server Activity

On this page:
- View Performance Metrics Compared to Baselines

Related pages:
- Dynamic Baselines
- Health Rules
- Hardware Resources Metrics

The Server Visibility module 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.
Servers List

On this page:
- Available Metrics in the Servers List

Related pages:
- Hardware Resources Metrics
- Troubleshoot Health Rule Violations

To access: AppDynamics Home > Servers

The Servers list shows all machines registered by a Standalone Machine Agent. This list shows metric data for Server-Visibility-enabled machines only. Machines registered with App or .NET Agents, with no Standalone Machine Agent running, do not appear 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 \textless\text{machine-path}\textgreater\textless\text{controller-info.xml}\textless\text{machine-path}\textgreater\textless\text{group}\textless\text{machine-path}\textgreater, then the name will be "myName". For \textless\text{machine-path}\textless\text{justName}\textless\text{machine-path}\textgreater, then the name will be "JustName".
- Filter the list using the dropdown to show all servers (the default) or just servers with Server Visibility enabled or disabled.
- Use View Options to select the columns shown for each server in the table. You can show or hide metrics such as CPU Steal, Load Average, Memory Swap Used, and Network I/O. You can also show or hide properties such as Host ID and JVM version.
- 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 the violations icon (⚠️) to view health rule violations for a server.
- Click on any point on a spark chart to see the percentage usage at that time.
- Sort the chart by any of the columns.
- Right-click on a set of selected servers to
  - Delete the servers from the Servers list. See "Delete Selected Machine Agent" in Manage Machine Agents.
  - View and analyze log data for the servers (if you have an Application Analytics license). See Application Analytics.
  - Set the Dynamic Monitoring Mode on the servers. See Dynamic Monitoring Mode and Server Visibility.
- See additional details of servers that are enabled for Server Visibility. Double-click a selected server to see server-specific dashboard, and details of the server's volumes, network, and processes.
- Filter data according to the "liveness" of servers. A Machine Agent that reports metrics from a server is considered live. Eventually, servers can transition to a terminated or unrecoverable state. By specifying a time range in the Server Dashboard, you can filter servers which were live and reporting metrics during that time range. To enable this option for an individual account:
  1. Log in to the Controller admin page (\textless\text{controller-hostname}\textgreater\textless\text{controller/admin.jsp}\textgreater)
  2. From Accounts, open the account page of interest.
  3. Click + Add Property and enter:
     a. Property = sim.server.pagination
     b. value = true
  4. Click Controller Settings > Controller Configurations and enter:
     a. Name = entity.liveness.filter.enabled
     b. value = true
     c. Name = entity.liveness.tracker.enabled
     d. value = true
- 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 (\textless\text{controller-hostname}\textgreater\textless\text{controller/admin.jsp}\textgreater)
  2. From Accounts, open the account page of interest.
  3. Click + Add Property and enter:
     a. Property = SIM_DISABLE_MACHINE_SORTING
     b. value = true
### Available Metrics in the Servers List

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>Specifies the root of the hierarchical path to the server. See the machine hierarchy property in this topic: <a href="#">Standalone Machine Agent Configuration Properties</a>.</td>
</tr>
<tr>
<td>Health</td>
<td>Indicates if server health rules have been violated on this machine.</td>
</tr>
<tr>
<td>Disk Usage % trend</td>
<td>The percentage usage trend over time of storage space in use across all listed volumes, partitions, and disks.</td>
</tr>
<tr>
<td>CPU (%)</td>
<td>The percentage of time the CPU was busy processing system or user requests.</td>
</tr>
<tr>
<td>CPU Trend</td>
<td>The trend over time of CPU usage.</td>
</tr>
<tr>
<td>Memory (%)</td>
<td>The percentage of memory used.</td>
</tr>
<tr>
<td>Memory Trend</td>
<td>The trend over time of memory usage.</td>
</tr>
<tr>
<td>Disk I/O (%)</td>
<td>The percentage of time spent performing read and write operations across one or more disks, volumes, or partitions.</td>
</tr>
<tr>
<td>Disk IO Trend</td>
<td>The trend over time of disk usage.</td>
</tr>
<tr>
<td>Network I/O (%)</td>
<td>The average network utilization of network bandwidth for all monitored network devices. The network utilization percentage is the total incoming and outgoing bytes per second, divided by the maximum speed of the network interface. In some cases, if the Network I/O percentage value is too small compared to the network speed (10,000 Mb/s), the value is rounded to zero on the Servers List. In this case, the Network I/O % shows as zero, while the drilled-down details of the machine might show network traffic.</td>
</tr>
<tr>
<td>Network Trend</td>
<td>The trend over time of network usage.</td>
</tr>
<tr>
<td>Server Visibility Enabled</td>
<td>Indicates if Server Visibility is enabled for this machine. Values=Yes, No, and Not Supported. Not Supported indicates that Server Visibility is not available for that machine OS.</td>
</tr>
<tr>
<td>Icon</td>
<td>Representation of the machine's OS</td>
</tr>
</tbody>
</table>
Server Dashboard

On this page:

- Server Dashboard Metrics
  - Host ID, Health, Hierarchy (Top Pane)
  - Load Average
  - Availability
  - CPU
  - Memory
  - Network
  - Volumes
  - Top 10 Processes Consuming CPU
  - Top 10 Processes Consuming Memory

Related pages:

- Hardware Resources Metrics
- Configure Health Rules to Monitor Servers
- Troubleshoot Health Rule Violations

To access: AppDynamics Home > Servers > Dashboard

Select a server from the list and double-click a server name. The Dashboard for the selected server appears. In this dashboard, 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.
- Find and switch to other Server Dashboards (pull-down menu next to server tier, top left).
- View an aggregate of the top 10 processes by CPU usage, and top 10 processes by memory.

Server Dashboard Metrics

Host ID, Health, Hierarchy (Top Pane)

- **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.
**Load Average**

The CPU load presented as an average over the past 1 minute, 5 minutes, and 15 minutes. The CPU load is the percentage of the CPU consumed by processes that are currently running or that are waiting to run. These metrics are shown as percentages in the Server Dashboard and are **scaled by 100 in the Metric Browser**.

**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.

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.
• **PPID**: Parent 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
Server Process Metrics

On this page:

- Metrics for Server Processes
- Windows Process Collection Configuration

Related pages:

- Hardware Resources Metrics
- Enable Server Visibility
- Machine Agent Settings for Server Visibility

To access: AppDynamics Home > Servers > double-click server > Processes

You can configure which processes AppDynamics monitors, how they are grouped, the number of processes to monitor, and how long they must be alive before monitoring them. For more information, see Machine Agent Settings for Server Visibility.

On the Server Processes tab you can:

- View all the processes active during the selected time period. The processes are grouped by class as 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.

### Windows Process Collection Configuration

When too many processes slow down the performance of the Machine Agent on Windows, you will need to adjust the system environment variables to accommodate. Specifically, in relation to Windows-based operating systems supported by the AppDynamics platform. See System Requirements.

Microsoft Windows process collection is enabled by default in the Controller and is missing the Effective User value due to performance issues. If you want to retrieve the Effective User value, set the `APPDYNAMICS_ENABLE_PROCESS_OWNER_INFORMATION` environment variable. Although you can retrieve this information, this will most likely slow your process collection and may lead to additional issues, such as high CPU usage and the failure to report metrics. To disable process collection altogether, set the `APPDYNAMICS_MACHINE_AGENT_STOP_PROCESS_COLLECTION`, which will speed up your performance at the expense of losing process metrics and metadata.

### Windows System Environment Variables

This information is specific to processes monitored on a Windows server.

1. **Default**: Retrieve all Windows processes and metadata. Starting with Machine Agent 4.5.4, Effective User information is excluded by default to improve performance.
2. `APPDYNAMICS_ENABLE_PROCESS_OWNER_INFORMATION`: Retrieve all Windows processes and metadata, including the Effective User, but the Machine Agent may experience performance issues. This is available starting with Machine Agent 4.5.4.
3. `APPDYNAMICS_MACHINE_AGENT_STOP_PROCESS_COLLECTION`: Stops all Windows process metrics and metadata collection, but improves Machine Agent performance. If this variable is set then the `APPDYNAMICS_ENABLE_PROCESS_OWNER_INFORMATION` variable is ignored. This is available starting with Machine Agent 4.4.0.

These environment variables only need to be set, you do not need to assign a value to the variable.

The Machine Agent will need to be restarted for these changes to take effect.
Server Volumes Metrics

On this page:
- Metrics in the Server Volumes Tab

Related pages:
- Hardware Resources Metrics

To access: AppDynamics Home > Servers > double-click server > Volumes

In the Server Volumes tab, you can:
- See in the list of volumes, the 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 in the Server Volumes Tab

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. (You can configure how this metric is calculated on Linux; see Configure Disk Usage Metric Collection on Linux.)
- **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 reads.
- **I/O Rate write**: The number of kilobytes per second of data writes.
- **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.
Server Network Metrics

On this page:

- Server Network Metrics

Related pages:

- Hardware Resources Metrics

To access: AppDynamics Home > Servers > double-click server > Network

In the Server Metrics tab you can:

- See the MAC, IPv4, and IPv6 address for each network interface.
- 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.

No metrics showing?
Network device speed cannot be found for some devices and some Linux versions.

Server Network Metrics

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.
Navigating Between Server and Application Contexts

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

With Server Visibility 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 Server health summary shows the number of critical/warning/normal conditions for the application’s servers.

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.
On the Servers Dashboard tab click APM Correlation.

Click on the APM Correlation link to go to the APM node associated with the server. If there is more than one APM node running on the server, a dialog box appears which allows you to select the desired APM node.
By selecting the APM node, you will be taken to the appropriate APM node Dashboard.
Configuring Docker Visibility

You can configure the Docker Visibility settings in the following file: `<machine-agent-home>/extensions/DockerMonitoring/conf/DockerMonitoring.yml`

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerProcessSelectorRegEx</td>
<td>The Machine Agent can monitor up to 150 running containers (depending on the pool size configuration, described below). The agent collects metrics from containers with one or more running processes whose command line matches a configurable regex. By default, the value of this regex is <code>.*</code> and matches all containers irrespective of the processes running in them. If you cannot restart the APM agents, or if you want to filter APM agents based on another command-line argument, you can override this behavior by editing this regex. Default = <code>.*</code></td>
</tr>
</tbody>
</table>
| Docker Metric Collector Pool Size | Specifies the local limit for the number of containers the agent can monitor. The effective limit for each agent is the minimum of  
  - The `sim.docker.machine.container.limit` on the Controller (default limit = 15 containers), or  
  - The Docker Metric Collector Pool Size on the agent (default pool size = 3, or 90 containers).  
You can change this setting to monitor up to 120 containers (pool size = 4) or 150 containers (pool size = 5). The trade-off is that increasing the pool size might lead to higher resource consumption on the agent host.  
To change the pool size, set the following.  
  - **System Property** · `appdynamics.docker.metric.collector.pool.size`  
  - **Environment Variable** · `APPDYNAMICS_DOCKER_METRIC_COLLECTOR_POOL_SIZE` |
| dockerTagsEnabled           | Enable (the default) or disable the collection of Docker tags from containers monitored by this agent.                                                                                     |
cgroupEnabled

The Standalone Machine Agent can collect Memory and CPU metrics for Docker containers using cgroupdata. This method is much faster than using the Docker API (the default metric-collection method). You can enable cgroupmetriccollection on all platforms except Amazon Linux OS. With the cgroup enabled flag set to true, the Standalone Machine Agent can monitor up to 600 containers per host. When cgroup is enabled, there's no need to update the Docker Metric Collector Pool Size. The max container limit is independent of this.

This mode is disabled by default. To enable cgroupmetriccollection, run the agent with the following command-line option set to true:

```
java -Dappdynamics.docker.container.collection.cgroup.enabled=true -jar machineagent.jar
```

Controller Settings for Docker Visibility

For information about how to configure these settings, see Controller Settings for Standalone Machine Agents.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim.docker.enabled</td>
<td>Enable the Docker Monitoring feature.</td>
<td>true</td>
</tr>
</tbody>
</table>
| sim.docker.machine.container.limit            | Global limit for the number of containers that each agent can monitor. The effective limit for each agent is the minimum of  
  * The sim.docker.machine.container.limit on the Controller (default limit = 15 containers), or  
  * The Docker Metric Collector Pool Size on the agent (default pool size = 3, or 90 containers).  
  You can specify this in the Administration Console as a Controller setting (all accounts) or as an Account setting for individual accounts. 150 is the maximum limit you can specify. | 15      |
| sim.machines.reuse.enabled                    | Reuse SIM Machine entities to handle the ephemeral environment.  
  Support is limited to Docker container machines for now.  
  If set to false, every new container will be considered a new machine. | true    |
| sim.docker.apmNode.markHistorical.enabled    | Mark the APM nodes as historical when a container running an app agent is stopped. | true    |
| sim.machines.tags.enabled                     | Enable or disable the import of Docker tags to the Controller.              | true    |
Dynamic Monitoring Mode and Server Visibility

You can selectively control the number of metrics reported by individual Machine agents (including Server Visibility agents). Rather than have all agents report all the metrics all the time, you can configure individual agents to collect

- Key Performance Indicator metrics only (KPI mode)
- KPI and Diagnostic metrics (Diagnostic mode)
- All available metrics (Advanced Diagnostic mode)

This provides the flexibility to report KPI metrics only on most machines and then increase the metric level on specific servers where you need deeper visibility to diagnose problems. You can increase scalability on the Controller and conserve metric bandwidth on the network with no sacrifice in visibility.

Every Basic and Server Visibility metric has a default Dynamic Monitoring Mode (DMM) class: KPI, Diagnostic, and Advanced Diagnostic. To see the DMM class for each metric, see Hardware Resources Metrics.

Important Notes

Note the following:

- You can disable DMM on individual agents. When DMM is disabled on an agent, that agent will report all metrics regardless of whether DMM is enabled or disabled on the Controller. Disabling DMM on an agent is recommended only for mission-critical servers and other machines for which you are sure you want to collect all metrics. For more information, see "Enable Dynamic Monitoring Mode (DMM)" under Standalone Machine Agent Configuration Properties.
- If you switch the monitoring mode in the Controller from a more-inclusive to a less-inclusive mode, the Metric Browser will show values for the newly-excluded metrics for one hour after the switch. For example: suppose you switch from Diagnostic to KPI mode. For any Diagnostic metric, the Metric Browser will report a steady line (at 0 or the last-reported value) for one hour after the switch; then the line will disappear. This is standard behavior in the Metric Browser for an agent when it stops reporting a specific metric.
- Each Machine Agent has a set of configurable settings that specify the volumes, networks, and processes to report and to ignore. For example, you can define a process whitelist (report on matching processes) and blacklist (ignore matching processes). An agent will not report metrics for items excluded by its local settings, regardless of the monitoring mode. For more information, see Machine Agent Settings for Server Visibility.
- If you have any custom health rules based on Diagnostic or Advanced Diagnostic metrics, DMM might cause these rules to generate "false-positive" alerts. (Standard, non-custom health rules are not affected.) If you have any health rules like this, the workaround is to edit the rule to use a KPI metric in place of the Diagnostic or Advanced Diagnostic metric.

Initial Setup

DMM is enabled by default. To configure DMM, do the following:

1. Log in to the Controller administration console using the root user password. For more information, see Access the Administration Console.

   http://<controller host>:[port]/controller/admin.jsp

2. Set the following options:
2. a. `sim.machines.dmm.defaultMode` = KPI (Sets the default mode to KPI on all Standalone Machine Agents).
   b. `sim.machines.dmm.dmmAllowed` = true (Enables Dynamic Monitoring Mode on the Controller)

3. For servers of interest that require additional visibility, increase the monitoring mode to Diagnostic or Advanced Diagnostic. To change the monitoring mode of one or more agents, do the following:
   - Log in to the Controller UI and click Servers.
   - In the Servers table, select the servers whose monitoring mode you want to change.
   - Right-click the selection, choose Select Dynamic Monitoring Mode and select the new mode.

Example Workflow

After you complete the initial setup, you can set the Dynamic Monitoring Mode on individual machine agents as needed. An example workflow might look like the following:

- The DevOps team for a large enterprise monitors its IT infrastructure using Standalone Machine Agents (1,000-plus agents monitoring servers in hundreds of locations). All agents are initially set to KPI mode.
- One agent reports a lot of disk read/write operations (KPI metric) on critical-server-A.
- Set the agent DMM on critical-server-A to Diagnostic.
- Monitor the amount of data read and written for the entire disk and for each partition (Diagnostic metrics) on critical-server-A.
- If the Diagnostic metrics do not indicate the source of the problem, and further investigation is needed, set the agent DMM to Advanced Diagnostic.
- Monitor average queue times and read/write times for each partition (Advanced Diagnostic metrics) on critical-server-A.
- When advanced diagnostics are no longer required on critical-server-A, set the agent DMM back to KPI.
Tier Metric Correlator

The Tier Metric Correlator enables you to identify load and performance anomalies across all nodes in a tier. Suppose you have a tier composed of a cluster of nodes running on containers or servers. You expect all the nodes to behave exactly the same under the same load conditions. How can you monitor this cluster for anomalies and outliers? The Tier Metric Correlator makes it easy to answer the following questions:

- Are all the nodes behaving within the expected band of performance, or do some nodes have outliers (slow calls, stalls, and errors)?
- In what time windows, and on which nodes, are these outliers occurring?
- Are outliers associated with a specific node cluster—for example, a cluster running a canary release?
- Are any resource issues (such as high CPU I/O or paging) correlated with these outliers?
- Are calls getting distributed evenly across all nodes in the tier?

This topic describes two example use cases:

- Using Tier Metric Correlation to quickly identify balances and imbalances in the distribution of calls across all nodes in a tier.
- Using Tier Metric Correlation to monitor and troubleshoot a canary deployment scenario. This example shows how you can easily compare performance across node clusters, identify nodes with transaction outliers, and see if any resource issues are causing these outliers.

Key Concepts

Transaction Outliers

The first step in metric correlation is to determine the transaction outliers for a tier: the number of Slow Calls, Very Slow Calls, and Stalled Calls whose response times are significantly outside the norm for that tier. The Transaction Outliers Heatmap visualizes the rate of these outliers and their distribution across all nodes in the tier.

A heatmap is a time-series chart with an extra dimension: the color intensity of each bar shows the distribution of outliers across all nodes. The darker the hue, the more nodes have outliers. The chart colors the bars in shades of gray (fewer outliers) and orange (more outliers).
Heatmaps make it easy to identify the normal bands of performance and any outliers for all nodes in a tier. In the following example, all nodes fall within two bands:

This heatmap highlights two time windows where the performance metric is noticeably higher for 50% of the nodes:

Correlated Metrics

The Correlated Metrics heatmaps enable you to correlate transaction outliers with specific resource metrics. The Correlate Metrics window makes it easy to identify performance metrics that correlate (and do not correlate) with transaction outliers of interest. The following example shows three heatmaps:

1. Transaction Outliers
2. Correlated metric
3. Uncorrelated metric (no outliers)
Troubleshooting Nodes and Servers

When you select a set of transaction outliers, the Selected Nodes and Impacted Servers charts show the distribution of these outliers across all nodes in the tier. This makes it easy to see if these outliers are associated with specific node clusters. Double-click on a pie chart to troubleshoot the node or server.
Example Use Cases

Comparing Load Distributions

In this example, a DevOps engineer wants to ensure that transaction calls are getting distributed evenly to all the nodes in a tier. She goes to the Tiers & Nodes view, right-clicks on a tier, and chooses Correlate Metrics. Looking at the Calls Per Minute heatmap, she can immediately see that the band of performance is 20-22 calls per minute, but for some nodes, the rate is higher during certain intervals. She decides to investigate the relevant load balancer and finds that a simple misconfiguration is causing the device to distribute calls unevenly at certain times. Using heatmaps, she can identify and fix a minor issue before it has a significant impact on her team's mission-critical applications.

Canary Deployment Testing

A DevOps engineer is responsible for a four-tier e-commerce application. The Order-Tier has five nodes running version 1.0 of the service. She deploys a "canary" (version 1.1 of the service) on one node. Before she deploys 1.1 on all nodes, she wants to see if there is any performance degradation on this node.
She opens the Controller, go to the **Tiers & Nodes** view for the application of interest, right-clicks on the Order-Tier, and chooses **Correlate Metrics**. The Tier Metric Correlator appears.

The **Transaction Outliers heatmap** shows that some calls are outliers: Errors, Slow Calls, Very Slow Calls, or Stalled Calls whose response times are significantly higher than the band of performance for that tier.

Her first question is: **Are these outliers associated with our “canary node” (ORD-N1)?** She drag-selects a set of these outliers. The **Node Distribution in Selection** pie chart (right) shows that all outliers are associated with the canary node. Clearly, the new code is not performing as well as the old code.
Her next question is: *Are any resource issues causing these outliers?* To answer this question, she examines the Correlated Metrics heatmaps to look for metrics that correlate with the outliers on the canary node. Most heatmaps show no correlation. For example, **CPU Busy%** shows that all nodes stay within the band of performance of 0-20%.

However, the CPU I/O Wait 95th Percentile(%) heatmap shows a strong correlation: All the metric outliers occur on the canary node, while all other nodes remain within the band of performance.

The **Pages paged out 95th Percentile (pages)** heatmap also shows a strong correlation with the transaction outliers on the canary node. With just a few clicks, she can immediately see that the canary node is performing worse; that the node has a CPU I/O problem; and that the CPU I/O problem is related to paging, which indicates a disk problem.
To reduce the visual noise and highlight the correlations, she unchecks all the uncorrelated metrics. Her next step is to investigate and troubleshoot the underlying server. To drill down into the canary node, she double-clicks on the Server Distribution in Selection pie chart (bottom right).

The Server Dashboard for the canary node appears. She goes to the Volumes tab and sees that a lot of spikes in I/O operations and queue wait times. She decides that the canary code is not ready to deploy to the entire tier. She needs to re-examine the canary code, fix the regression, and re-test.
Enabling Tier Metric Correlation

Tier Metric Correlation must be enabled per account on the Controller. If you are using a SaaS Controller, contact AppDynamics Support. If you are using an on-premises Controller, do the following:

1. Log in to the Controller administration console using the root user password:
   http://<controller host>:<port>/controller/admin.jsp
2. Go to the Accounts page and select the account for which you want to enable this feature.
3. Click Add Property in the accounting settings and add the following:
   
   ```
   = ENABLE_SIM_HEATMAPS = true
   ```

To correlate percentile metrics, you must enable percentile metric reporting on both the Controller and the Machine Agent. By default, reporting is disabled on the Controller and enabled on the agent.

- To enable/disable reporting on the Controller, log in to the Controller administration console and set the `sim.machines.perc entile.percentileMonitoringAllowed` property. See Controller Settings for Standalone Machine Agents.
- To enable/disable reporting on the agent, open the `<machine_agent_home>/extensions/ServerMonitoring/conf/ServerMonitori ng.yml` file and set the `percentileEnabled` property. See Machine Agent Settings for Server Visibility.

Workflow Description

The following steps outline the general workflow:

- Go to the Tiers & Nodes view for the application of interest.
- Right-click on the tier and choose Correlate Metrics. The Tier Metric Correlator appears.
- Identify **transaction outliers** for the tier – calls flagged as Slow, Very Slow, Stalls, or Errors (see **Business Transaction Performance**).
- Drag the cursor to select the outliers of interest. The pie charts on the right show the distribution of these outliers by node and server.
- Identify any **correlated metrics** for the outliers. Unselect checkboxes for non-correlated metrics.
- Identify and troubleshoot the nodes and servers where the outliers are occurring. Double-click on a pie slice to open the dashboard view. You can use the correlated metrics to guide your troubleshooting.
Service Availability Monitoring

To access: AppDynamics Home > Servers > Service Availability (left navigation menu)

The Service Availability feature of Server Visibility enables you to monitor internal or external HTTP and HTTPS services. You configure service monitoring from the controller UI. After you configure monitoring for a service, Service Availability evaluates the service as follows:

- The agent evaluates each response, based on your specified violation rules, and flags each response as failed (rule violation) or successful (no violation).
- The service monitor maintains a rolling buffer of evaluated responses. This buffer has a configurable window size (number of evaluated responses), success threshold (number of successful responses), and failure ratio (number of failed responses).
- The monitor collects evaluated responses until the response buffer is full. Then it evaluates the service as:
  - NORMAL - Successful responses are greater than or equal to the success threshold.
  - CRITICAL - Failed responses is greater than or equal to the failure threshold.
- When the buffer is full, the monitor re-evaluates the service every time it evaluates a new response.

For example, with the 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.

Licensing and System Requirements

Service Availability Monitoring requires a separate license and a Server Visibility license. Server Visibility is currently available for Linux, Windows, and Solaris. See License Management and Server Visibility Requirements and Supported Environments.

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.

### Service Availability

<table>
<thead>
<tr>
<th>Monitored Service</th>
<th>Server</th>
<th>State</th>
<th>Success Rate (%)</th>
<th>Response Time (ms)</th>
<th>Response Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ecommerce Homepage</td>
<td>ECommerce_MA</td>
<td>✔</td>
<td>100.0</td>
<td>3</td>
<td>5.0k</td>
</tr>
<tr>
<td>Ecommerce Marketing Landing Page</td>
<td>ECommerce_MA</td>
<td>✗</td>
<td>0.0</td>
<td>2</td>
<td>1.5k</td>
</tr>
</tbody>
</table>

### Monitored Service Details

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

### 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 configure this feature, users need the **Configure Server Visibility** permission. See **Server Visibility Permissions** for more details.
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.

   ![Add Service Monitoring Configuration Window](image)

   - **Main** tab
   - **Request Configuration** tab
   - **Response Validators** tab

3. On the Main tab, specify the required fields and save the configuration. Each field has an associated tooltip 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, specify 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, <a href="http://myThirdPartyService.com/data">http://myThirdPartyService.com/data</a>. 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 enabled for Server Visibility 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. <strong>Note</strong> - Setting this interval to more than 60 seconds will result in visible gaps in the metric graphs for monitored services.</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>
</tbody>
</table>
### 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 Visibility 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 Actions section of the Policy window, add the action you want to execute if the policy violates and save the policy.
Server Visibility Events

To access: top navigation bar > Servers > Events

You can see Machine Agent events on the Server Events list and on the Application Events list for applications associated with the machine agent. To access the Application Events list, see Monitor Events.

Machine Agent events include the following:
- Custom Events
- Agent Internal Diagnostic Events

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.
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
Machine Agent Settings for Server Visibility

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

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


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.
- The agent updates dynamically in response to agent configuration property changes, so there is no need to restart the agent after you update this file.

Configurable Server Visibility Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>basicEnabled</td>
<td>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</td>
</tr>
<tr>
<td>volumeMonitorConfig:</td>
<td>Do not report more than N volumes, where N = maxNumberVolumes. 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. Default = 5</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>volumeMonitorConfig:</td>
<td></td>
</tr>
<tr>
<td>whitelistSelectorRegex</td>
<td></td>
</tr>
<tr>
<td>blackistSelectorRegex</td>
<td></td>
</tr>
<tr>
<td>samplingInterval</td>
<td></td>
</tr>
<tr>
<td>networkMonitorConfig:</td>
<td></td>
</tr>
<tr>
<td>maxNumberNetworks</td>
<td></td>
</tr>
<tr>
<td>whitelistSelectorRegex</td>
<td></td>
</tr>
<tr>
<td>blacklistSelectorRegex</td>
<td></td>
</tr>
<tr>
<td>defaultProcessClassSelector</td>
<td></td>
</tr>
</tbody>
</table>
| **processClassSelectorRegexList** | 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 default `ProcessClassSelector regex` to the command line.
3. If there is no match for the default `ProcessClassSelector 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.

<p>| <strong>samplingInterval</strong> | Indicates how often to gather metric data. Units in milliseconds. Default = 60000 (60 seconds) |
| <strong>maxClassIdLength</strong> | 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 |
| <strong>processSelectorRegex</strong> | 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 regular expression will filter out any processes where the command line ends with a close bracket (<code>]</code>). For Linux, this means that the process arguments could not be found and usually, that is a kernel process. For Windows, no processes should end with a bracket character, so consequently, the regex should include all processes on Windows. Default = &quot;^^-]$&quot; |
| <strong>minLiveTimeMillisBeforeMonitoring</strong> | 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) |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxNumberMonitoredClasses</td>
<td>Specifies the maximum number of process classes that the machine agent monitors. The processes that are reported are the processes using the highest CPU and memory that match the regex specified by processSelectorRegex. Default = 20</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Changing this setting can affect the resource consumption of your deployment. Before you change this setting verify your application environment and Controller can handle the increased resource requirements.</td>
</tr>
<tr>
<td>defaultDiskSectorSize</td>
<td>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</td>
</tr>
<tr>
<td>memoryMonitorConfig</td>
<td>Specifies a custom sampling interval for collecting memory metrics on Linux: Default = 3000</td>
</tr>
<tr>
<td>cpusMonitorConfig</td>
<td>Specifies a custom sampling interval for collecting CPU metrics on Linux: Default = 3000</td>
</tr>
<tr>
<td>tag</td>
<td>A list of user-defined tags for the individual server. You can use these tags to query, filter, aggregate, and compare related servers. See Server Tagging. Each tag is specified by a key-value pair. You can define tag names up to 127 unicode characters and tag values up to 255 unicode characters. Define each key on a separate line. All key/value strings should be within single quotes. If a key has multiple values, delineate the list with commas:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;key&gt;</code>: [ <code>&lt;value&gt;</code> ] \n</td>
</tr>
</tbody>
</table>

**Docker Visibility Settings**

For information about Docker Visibility settings on the Standalone Machine Agent, see Configuring Docker Visibility.

**Process Limits**

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

- Total number of processes displayed in the UI for a single query is 5000 processes per call.
- Total number of processes tracked per account. The default value is 10000 processes per account.

For more details, see Controller Settings for Server Visibility.
Configure Health Rules to Monitor Servers

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

Related pages:
- Health Rules
- Alert and Respond
- Policies
- Health Rules
- Actions

You can configure Server Visibility 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 Visibility. You can modify these and set up your own health rules. Choose Alert & Respond > Health Rules to see a list of the rules on the Controller.

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 Visibility

This topic describes Controller Admin settings that are specific to Server Visibility. You can apply these settings and permissions from the Account tab in the Controller Administration UI. You need the root user password to change these settings. Changes to the settings take effect the next time the agent is restarted and connects to the Controller. Not all configuration properties can be set at the account-level, check the Description column for each property to determine whether a specific Server Visibility Property can be enabled. For more about roles, see Roles and Permissions and Manage Users and Groups.

For details on how to change the Controller Admin settings for machine agents, see Controller Settings for Standalone Machine Agents.

<table>
<thead>
<tr>
<th>Server Visibility Property</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim.docker.enabled</td>
<td>Enable Docker Visibility.</td>
<td>true</td>
</tr>
<tr>
<td>sim.docker.machine.container.limit</td>
<td>Global limit for the number of containers to monitor per machine. You can specify this in the Access the Administration Console as a Controller setting (all accounts) or as an account setting for individual accounts. The maximum limit you can specify is 120 containers.</td>
<td>100</td>
</tr>
<tr>
<td>sim.docker.monitorAPMContainersOnly</td>
<td>When set to true, allows you to monitor APM containers only. If the value is set to false, monitors all the containers on the machine except the machine agent container. This property can be applied at an account level and enables the monitoring of containers with and without APM. This property was formerly named sim.docker.infraMode.enabled.</td>
<td>true</td>
</tr>
<tr>
<td>sim.exceptions.stacktrace.enabled</td>
<td>When this setting is enabled, controller stack traces are sent in the error response when a client request encounters an error. Setting this to true is a security risk, it should only be set to true if you understand the consequences and if directed by a support engineer.</td>
<td>false</td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| `sim.machines.agent.process.maxClasses` | 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 Machine Agent Settings for Server Visibility. |
| `sim.machines.count.maxPerAccount` | 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. |
| `sim.machines.deleteStaleMachines.maxLimit` | Set the number of machines to be deleted every 10 minutes, which is one cycle of the purger. For better performance, you can purge up to 700 machines per cycle. |
| `sim.machines.dmm.defaultMode` | Sets the Dynamic Monitoring Mode and Server Visibility default on the Controller. The allowable settings are:  
- **KPI** – Collect Key Performance Indicator metrics only  
- **DIAGNOSTIC** – Collect KPI and Diagnostic metrics  
- **ADVANCED_DIAGNOSTIC** – Collect All available metrics  
*You must enter one of these exact strings. If you enter a different value (such as "advanced-diagnostic"), the Controller will not update the setting and some servers might not appear in the Servers list.* |
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim.machines.dmm.dmmAllowed</td>
<td>With this option enabled, the Controller collects metrics from each Standalone Machine Agent based on the Dynamic Monitoring Mode and Server Visibility setting on that agent. If this option is disabled, the Controller collects all available metrics from each agent.</td>
<td>false</td>
</tr>
</tbody>
</table>
| sim.machines.dotNet.HostIdConflictCheckAllowed | With this option enabled, the Controller throws an exception if a Standalone Machine Agent tries to register with the Controller while both of the following conditions are true:  
  1. A .NET agent on the same machine is already registered with the same <host-id> specified for the Standalone Machine Agent, and  
  2. .NET Compatibility Mode is disabled on either the Controller or the Standalone Machine Agent.  
  The generated exception acts as a reminder that that .NET Compatibility Mode mode must be enabled on both the Controller and the Standalone Machine Agent.  
  **Note:** Do not disable this option unless instructed by AppDynamics Support. | true        |
<p>| sim.machines.hostidMappingAllowed           | 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. | false       |
| sim.machines.offline.toStaleTimeoutMillis   | 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 it means less history. | 2592000000 ms (30 days) |
| sim.machines.registrations.hashCache.enabled | Prevent subsequent registration requests from fetch data and compare if nothing has changed in the request payload. | true        |</p>
<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sim.machines.registrations.maxPerSecondPerAccount</code></td>
<td>Set the 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 to 600/sec. A suggested formula is ((\text{number of Machine Agents, version 4.2 or higher} / 60)). 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. Changing this setting can affect the resource consumption of your deployment. Before you change this setting, verify your application environment and Controller can handle the increased resource requirements.</td>
<td>60</td>
</tr>
<tr>
<td><code>sim.machines.registrations.update.frequency</code></td>
<td>Update machines' timestamp on a specific frequency, regardless of stale timeout value. This allows the purger to use only metadata for purging machines.</td>
<td>86400000ms (1 day)</td>
</tr>
<tr>
<td><code>sim.machines.reuse.enabled</code></td>
<td>Reuse SIM Machine entities to handle ephemeral environment. Support is currently limited to Docker container machines. If set to false, every new container will be considered a new machine.</td>
<td>true</td>
</tr>
<tr>
<td><code>sim.machines.simAllowed</code></td>
<td>This setting allows you to enable or disable Server Visibility from the controller.</td>
<td>true</td>
</tr>
<tr>
<td><code>sim.machines.stale.purgeIntervalMillis</code></td>
<td>An interval in milliseconds that determines when stale machines are deleted from the Controller database. If the duration of 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>21600000ms (6 hours)</td>
</tr>
<tr>
<td><code>sim.machines.tags.maxPerAccount</code></td>
<td>The maximum number of unique tags per account.</td>
<td>1000</td>
</tr>
<tr>
<td><code>sim.machines.tags.maxPerMachine</code></td>
<td>The maximum number of tags per machine.</td>
<td>50</td>
</tr>
<tr>
<td><code>sim.metrics.metricBrowser.machineMetricMappings.enabled</code></td>
<td>Enables or disables the Metric Browser to display the metrics reported per machine. When this feature is enabled, tier level aggregated metrics, available under Tier node in the Server metric browser, are not visible. With Streamlined Browsing enabled, the Hardware Resources and Custom Metrics folders are not displayed.</td>
<td>true</td>
</tr>
<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 | 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).  
  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.metrics.trackRegisteredEnabled         | Enable Streamlined Browsing for node metrics. This option is useful when you are browsing metrics for tiers that contain multiple nodes.                                                                      | false |
| sim.perAccountProperty.enabled            | Enables a user to configure properties at an account level.                                                                                                                                                 | true  |
| sim.perAccountProperty.syncFrequencyInMillis | The interval in milliseconds for the per Account Property data to sync with a database.                                                                                                                      | 300000|
| sim.perAccountProperty.syncInBackground   | Enables syncing the per Account Property data with a database in the background. For better performance, set this value to true.                                                                          | true  |
| sim.processes.bulkDelete.enabled          | Delete processes in bulk instead of fetching one by one. For better performance, set this value to true.                                                                                                  | true  |
| sim.processes.delete.maxCount             | Set the maximum number of processes to be deleted every 10 minutes, which is one cycle of the purger. For better performance, set the delete maxCount processes value to 55000 per cycle.                                | 10000 |
| sim.processes.count.maxPerAccount         | The maximum number of processes stored per account.                                                                                                                                                         | 300000|
| sim.processes.count.maxPerMachine         | The maximum number of processes that each Machine Agent can monitor.                                                                                                                                       | 1000  |

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.
| sim.processes.creation.maxConcurrent | 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 process 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 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 > 5000). So, Controller can only process one request, and rejects the other one.

Case 3: Two Machine Agents (MA1 and M2) each try to register 500 processes and the limit is set to 5000. Both requests arrived at the Controller and since 500 + 500 < 5000, then both requests are processed and the processes are registered. | 5000 |
| sim.processes.query.maxResultLimit | The maximum number of processes that the UI can show (for example, in the Server Dashboard > Processes tab). Suppose the time range is 2 weeks and the agent reported over 10000 processes during that interval. If maxResultLimit = 5000, the UI will show 5000 processes. | 5000 |
| sim.processes.registrations.maxPerSecondPerAccount | Set the maximum number of process requests handled per second per account. For better performance, set to 600/sec. On a multi-tenant Controller the total number of agent per sec should not exceed 600/sec. | 60/sec |
| sim.processes.stale.purgeIntervalMillis | Number of milliseconds between consecutive deletes of stale processes for an account. | 3600000 ms (1 hour) |
| sim.processes.terminated.toStaleTimeoutMillis | Number of milliseconds before a terminated process is considered stale and can be deleted to make space for new data. | 172800000 ms (2 days) |
Machine Agent Hierarchy

To group servers together so that health rules can be applied to the specific server groups, use the Machine 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 Visibility dashboard.

You need a Server Visibility license to use this feature.

You can specify the Machine Hierarchy property using controller-info.xml, a system property, or an environment variable.

To achieve something like this where you group servers into geo locations, perhaps representing different data centers:

```
<machine-path>AD-Financial|Node1</machine-path>
```

Metric Browser view for this example:
When creating health rules, on the **Affected Entities** tab you can select one or more subgroups.
Server Tagging

Server tagging makes it easy to query, filter, and compare related servers using custom metadata. You can tag related servers based on OS, location, tier, owner, or any other relevant criteria. Server tags provide additional context to server metrics. For example, you might want to specify deployment-version tags to the servers in a specific cluster; then you can use these tags to identify anomalies in server metrics during a new deployment.

Service Tagging requires a Server Visibility license. Server Tagging is currently available for Linux, Windows, and Solaris.

---

Server Visibility can import the following tags automatically.

<table>
<thead>
<tr>
<th>Source</th>
<th>Tags</th>
<th>Controller / Agent Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom</td>
<td>Server Visibility can import</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Custom tags defined in ServerMonitoringConfig.yml, as described in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defining and Viewing Tags for a Server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Custom tags defined in Amazon EC2 (search for Tagging EC2 Resources</td>
<td></td>
</tr>
<tr>
<td></td>
<td>in the Amazon AWS documentation).</td>
<td></td>
</tr>
<tr>
<td>Machine</td>
<td>These tags appear in the Server Dashboard under Tags.</td>
<td></td>
</tr>
<tr>
<td>Agent</td>
<td>- OS Architecture</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- OS Kernel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- AppDynamics Agent Version</td>
<td></td>
</tr>
<tr>
<td>Amazon</td>
<td>These tags appear in the Server Dashboard under Tags as AWS</td>
<td>tag</td>
</tr>
<tr>
<td>Web Services</td>
<td>Tags from Amazon Web Services below.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- resource-id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- availability-zone</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- region</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- ami-id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- instance-type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- security-group</td>
<td></td>
</tr>
<tr>
<td>Docker</td>
<td>Server Visibility can import user-defined tags and system container-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>level tags. These appear under Container Details &gt; Tags as Docker</td>
<td>tag</td>
</tr>
<tr>
<td></td>
<td>- The exact set of tags imported can vary depending on the system on</td>
<td></td>
</tr>
<tr>
<td></td>
<td>which the container is running.</td>
<td></td>
</tr>
</tbody>
</table>

---

*Custom tags defined in ServerMonitoringConfig.yml, as described in Defining and Viewing Tags for a Server below.*

*Custom tags defined in Amazon EC2 (search for Tagging EC2 Resources in the Amazon AWS documentation).*

*These tags appear in the Server Dashboard under Tags.*

*These tags appear in the Server Dashboard under Tags as AWS |tag|. See Importing Tags from Amazon Web Services below.*

*Server Visibility can import user-defined tags and system container-level tags. These appear under Container Details > Tags as Docker |tag|. The exact set of tags imported can vary depending on the system on which the container is running.*
Kubernetes  
OpenShift

Server Visibility can import ReplicaSet and pod tags. These appear under Container Details > Tags as K8s|<tag>.

- Controller 4.4.3 or higher
- Standalone Machine Agent 4.4.3 or higher

Viewing Servers by Tag in the Controller UI

View filter servers by tag in the Servers list, click the Filters button and add the criteria you want for the tag filer. If you specify multiple criteria, the filter performs an AND search (for criteria with different keys) and an OR search (for criteria with the same key). In the following example, the tag criteria filters all servers with

1. A Tier tag that equals ECommWeb OR ECommInventory, AND
2. An OS Architecture tag that equals x86_64.

Defining and Viewing Tags for a Server

You can specify custom tags as a set of key/value strings in the agent config directory. Each agent can support up to 50 tags by default.

Note the following good practices for defining server tags:

- If an agent is deployed in Amazon Web Services, the agent can auto-detect and import tags defined in AWS. The Controller also auto-assigns a set of default tags to each server.
  
  *The primary use case for custom tags is to specify information that is not already specified in AWS and auto-assigned tags.* Before you specify custom tags in the agent config directory, it is good practice to review the current set of tags: go to the Server Dashboard and view the Tags pane at the bottom of the page.
Configure the following option in `<machine_agent_home>/extensions/ServerMonitoring/ServerMonitoringConfig.yml`:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| tag          | A list of user-defined tags for the individual server. You can use these tags to query, filter, aggregate, and compare related servers. Each tag is specified by a key-value pair. You can define tag names up to 127 Unicode characters and tag values up to 255 Unicode characters. Define each key on a separate line. All key/value strings should be within single quotes. If a key has multiple values, delineate the list with commas: `<key>: [ <value> ]`

```
tags:
  'Location': ['NYC', 'Data Center', 'Server Room 7']
  'Environment': ['preProduction']
```

Importing Tags from Amazon Web Services

Detailed information about Amazon Web Services is outside the scope of AppDynamics documentation. The following links are provided as a convenience and might not work if Amazon reorganizes its website or documentation.

Before Server Visibility can import tags from AWS, you must set up IAM roles:

1. Log in to the AWS console: https://console.aws.amazon.com/iam/
2. Create a role that has read access to EC2 tags. Specifically, the role must have an AWS Managed Policy with the required permissions (such as AmazonEC2ReadonlyAccess) attached.
3. Add this role to your EC2 instance. For information about these two steps, search for "To create an IAM role using the IAM console" and "Attaching an IAM Role to an Instance" in the following page:
4. Run the Machine Agent with Amazon Web Services enabled.

Configuration Options for Server Tagging

To edit these settings, log in to the Controller administration console using the root user password. See Access the Administration Console.

<table>
<thead>
<tr>
<th>Server Visibility Property</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim.machines.tags.enabled</td>
<td>Enable/disable server tags for all servers.</td>
<td>True</td>
</tr>
<tr>
<td>sim.machines.tags.aws.enabled</td>
<td>Enable/disable the collection of AWS tags for all servers.</td>
<td>True</td>
</tr>
<tr>
<td>sim.machines.tags.aws.pollingInterval</td>
<td>Length of time in milliseconds between each polling of AWS tags</td>
<td>21600000 (6 hours)</td>
</tr>
<tr>
<td>sim.machines.tags.maxPerMachine</td>
<td>Maximum number of unique tags per account.</td>
<td>50</td>
</tr>
<tr>
<td>sim.machines.tags.maxPerAccount</td>
<td>Maximum number of unique tags per account.</td>
<td>500</td>
</tr>
</tbody>
</table>

Configuration Options for Docker Tags

The machine agent can be configured to collect different types of tags. By default, all tags are collected. To turn off tag collection, set dockerTagsEnabled to "false".

2. Under the containerMonitoringConfig section, set dockerTagsEnabled: "false"
# WARNING: Before making any changes to this file read the following section carefully
#
# After editing the file, make sure the file follows the yml syntax. Common issues include
# - Using tabs instead of spaces
# - File encoding should be UTF-8
#
# The safest way to edit this file is to copy paste the examples provided and make the
# necessary changes using a plain text editor instead of a WYSIWYG editor.
#
# samplingInterval indicates how often to gather metric data. Units in milliseconds.
samplingInterval: 30000

containerMonitoringConfig:
  # containerProcessSelectorRegex defines regular expression to evaluate the processes in
  # each running container to be monitored by the machine agent.
  # The regular expression is compared against each process full command line within running
  # If the pattern matches, then the machine agent start monitoring it.
  containerProcessSelectorRegex: ".*[ ]-Dappdynamics.*"
dockerTagsEnabled: "false"

Configuration Options for AWS Tags

The machine agent can be configured to collect different types of tags. By default, all tags are collected. To turn off tag collection, set awsTagsEnabled to "false".

2. Set awsTagsEnabled: "false"
Configuration Options for Kubernetes and OpenShift Tags

The machine agent can be configured to collect different types of tags. By default, all tags are collected. To turn off tag collection, set `k8sTagsEnabled` to "false".

2. Set `k8sTagsEnabled: "false"`
//ServerMonitoring.yml
# WARNING: Before making any changes to this file read the following section carefully
#
# After editing the file, make sure the file follows the yml syntax. Common issues include
# - Using tabs instead of spaces
# - File encoding should be UTF-8
#
# The safest way to edit this file is to copy paste the examples provided and make the
# necessary changes using a plain text editor instead of a WYSIWYG editor.
#
k8sTagsEnabled: "false"
# samplingInterval indicates how often to gather metric data. Units in milliseconds.
samplingInterval: 30000
...
Tagged Metrics

The Tagged Metrics window enables you to aggregate all nodes in a tier based on a server tag (user-defined, or imported from AWS or Kubernetes) and then compare performance metrics between the node groups in one set of charts. For example, you can quickly compare the performance on all "blue nodes" and "green nodes" during a canary deployment.

1. In the Controller left-side menu, choose More > Tagged Metrics.
2. In the Tagged Metrics window, select the server tag and the tier of interest.
3. Compare the response times, call rates and transaction outliers (errors, slow calls, very slow calls) between the tag-based node groups.

The following chart shows how you can easily compare performance between different sets of nodes. The user wants to analyze node performance for a tier with 26 nodes running v1 of the software and one node running v2. The user selects the "version" tag, which represents a Kubernetes label. The charts show very clearly that the rate of slow and very slow calls has increased dramatically on the v2 node.

This feature is disabled by default. To enable:

2. In Controller Settings, search for "tagged.metrics.enabled" and set this field to true.
3. In Account Settings:
   a. Select the account to enable.
   b. Under Account > Additional Properties: click Add Property, add the "tagged.metrics.enabled" property, and set it to true.
Network Visibility

With Network Visibility, you can

- Answer the key question quickly and easily: *Is the network to blame for any performance issues in my application?*
- Identify the root cause of performance issues in the application, the network, or in application/network interactions (i.e., how an application or server utilizes the network).
- Pinpoint the traffic flows, individual nodes, and transport connections where network or application/network issues are occurring.
- Collect detailed metrics that show how the network is performing and how well your application uses network connections and resources.
- Collect and report targeted troubleshooting information to network, IT, DevOps, and other teams.

---

**Installation and Administration on Linux**

- Set Up Network Visibility on Linux
- Set Up the Network and App Agents on Linux
- Administer Network Agents on Linux

**Installation and Administration on Windows**

- Set Up Network Visibility on Windows
- Install the App Agent on Windows
- Administer Network Agents on Windows

**Using Network Visibility**

- Managing Network Agents in the Controller
- Monitor Network Visibility Metrics
- FAQs and Troubleshooting for Network Visibility
- Advanced Operations

**Reference**

- Network Visibility Concepts
- Workflows and Example Use Cases
- User Interface
- Network Visibility Events
Network Visibility Overview

Application Intelligence and Network Visibility

Network Visibility extends the application intelligence of AppDynamics APM down the stack from the application to the network. With "app-only" visibility, it can be easy to mistakenly blame (or not blame) the network when an application issue arises. Network Visibility can help reduce or eliminate the guesswork involved in identifying root causes. Network Agents and App Agents, working together, automate the work of mapping TCP connections to the application flows that use them. Network Agents can identify intermediate load balancers (which often split TCP connections) and correlate the connections on either side of these devices.

The agent-based approach of Network Visibility has the following advantages over standard approaches to network monitoring:

- More cost-efficient than using network monitoring appliances, which often view traffic from a few central locations
- Especially useful for distributed environments and multi-tier applications that span multiple network segments
- Works in cloud and hybrid networks, unlike most network-monitoring solutions

Drilling Down to the Root Cause

If network issues are affecting your application, Network Visibility can help you determine the even a minor cause.

- You see a spike in transaction outliers in the Application Dashboard. Are network issues to blame?
- Switch over to the Network Dashboard. Each tier, node, and link shows network KPIs (Key Performance Indicators) that measure the network health of that element. Turn on baselining to highlight network elements with KPIs outside the baseline.
- To diagnose a tier, node or network link, right-click and choose View Metrics. In the right-click dashboard, look for network metrics with spikes that correlate with the spikes in your transaction outliers. This is often all you need to do to find the network root cause.
- If a network element requires more in-depth troubleshooting, configure the relevant Network Agents to collect metrics on the individual Connections used by that element. You can then
  - Click on a node or link and view KPIs for the Connections used by the relevant nodes.
  - Right-click on a Connection and view detailed metrics in a right-click dashboard or in the Metric Browser.

Network Visibility Metrics

Network Visibility collects and displays the following types of metrics:

- Network KPIs (Key Performance Indicators) provide high-level, at-a-glance measures of whether the network is affecting the performance of the monitored application. The Network Flow Map shows KPIs for each tier, node, and link.
- The PIE (Performance Impacting Events) metric enables you to see immediately if there are any such events on a connected client, server, or network link.
If the KPI metrics indicate an issue with a specific element, you can view additional metrics for that element to identify root causes. Simply right-click on an element and choose View Metrics. The metrics and charts in the right-click dashboard are all designed to answer the question: Are there any bottlenecks on this element that are affecting my applications?

To perform in-depth analysis, you can view detailed TCP Flow metrics in the Metric Browser. You can view node metrics to evaluate the health of TCP sockets and network interfaces.

**Network Visibility for Multiple App Nodes on the Same Server**

Network agents can monitor multiple nodes that are associated with the same IP address because they run on the same physical or virtual server. The agent monitors each node individually and calculates network metrics for each node. These metrics are based on the ingress/egress traffic for each individual node, not aggregate traffic for the IP address of the host on which the node is running.
Network Visibility Requirements and Supported Environments

On this page:
- System Requirements
- Linux System Requirements
- Network Visibility on Linux
- Windows System Requirements
- Network Visibility on Windows
- Limitations
- Known Issues

Related pages:
- License Management
- Java Supported Environments

Please review the following guidelines and notes before you enable Network Visibility.

System Requirements

Network Visibility requires a supported version of the Network Agent to be installed on the monitored server.

Network Visibility features are available for Linux and Windows. You need a Network Visibility license to enable and use Network Visibility features. Network Visibility is enabled by default on AppDynamics controllers. For more information, see License Entitlements and Restrictions.

Linux System Requirements

Network Visibility requires:
- Controller version 4.4 or higher with a Network Visibility license.
- You must have sudo or root access permissions on the agent host to install the Network Agent. You do not need sudo or root access to run the agent.
- Each host on Linux must have
  - A Network Visibility Agent (version 4.4 or higher)
  - A Java App Server agent (version 4.4 or higher) with an App Agent license
- Network Visibility is supported on Linux hosts with Java App Server Agents only.

Network Visibility on Linux

The Network Agent is supported on Linux distributions based on glibc 2.12 or higher:
- CentOS 6 and higher (32-bit and 64-bit)
- Ubuntu 14 and higher (32-bit and 64-bit)
- Red Hat Enterprise Linux 6 and higher (32-bit and 64-bit)
- Fedora 24 and higher (32-bit and 64-bit)

The Network Agent host must have the following libraries pre-installed:
- libdl.so.2
- libpthread.so.0
- librt.so.1
- libm.so.6
- libc.so.6
- libncurses.so.5
- libtinfo.so.5
- libstdc++.so.6
Windows System Requirements

Each Windows host must have:

- A Java agent, version 4.5.4.24386 or later, with an App Agent license. See Install the Java Agent and License Management.
- A Controller, version 4.5.4.24386 or later, compatible with the Java Agent. See Agent and Controller Compatibility. Installing AppDynamics to test or evaluate, typically involves verifying system requirements, preparing the host, and then performing the Controller installation. See Install the Controller Using the Enterprise Console and Install the Controller Using the CLI.
- The Network Visibility Agent. Download the Network Agent version 4.5.7 or higher.
- An account with administrative privileges on the Windows machine where you want to install the Network Agent.

Network Visibility on Windows

The Network Agent is only supported for Java applications on these Windows distributions:

- Windows Server 2019 (64-bit)
- Windows Server 2016 (64-bit)
- Windows Server 2012 (64-bit)
- Windows Server 2012 R2 (64-bit)
- Windows Server 2008 (64-bit)
- Windows Server 2008 R2 (64-bit)

**Note**
Network Visibility on Windows supports only Java applications.

Limitations

- This release does not support the following Network Flow Map features:
  - Federated Flow Maps
  - Visualization of flows between web servers and APM entry tiers
  - The Network Flow Map does not filter out connections based on the selected time range.
- **The Network Agent cannot monitor multiple App Server Agents on the same host if these agents report to different Controllers. All App Server Agents must report to the same Controller.**
- **The Network Dashboard does not show data for Health Rules.**
- **You cannot view Health Rule violations in the Network Dashboard. To view any Health Rule violations, including those based on Network Visibility data, go to the Application Dashboard.**

Known Issues

- If you open the Network Flow Map for an individual node, the KPI metrics for node-to-load-balancer and node-to-TCP-endpoint links show network KPIs for all nodes in the parent tier (instead of KPIs for the individual node only). To view KPIs for the individual node, open the link popup and look at the Connection KPIs.
- **Network Visibility cannot monitor Cross Application Flows that use Jersey web servlets. As a workaround, you can disable Jersey servlet instrumentation. The impact is that Business Transactions on these flows get recognized as servlets instead of Web Services. To implement this workaround:**
  1. Select the application of interest in the Controller UI.
  2. Choose Configuration > Instrumentation > Transaction Detection.
  3. Disable instrumentation for Jersey Servlet and Jersey 2.x Servlet

See also Network Visibility Known Issues in the Release Notes.
## Set Up Network Visibility on Linux

### Related pages:
- ZIP Network Agent Operations
- RPM Network Agent Operations
- DEB Network Agent Operations
- Network Agent Rules for AppDynamics Universal Agent
- Docker and Network Visibility

The following table outlines the required steps for setting up Network Visibility on an individual host.

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Review the <strong>Network Visibility Requirements and Supported Environments</strong></td>
</tr>
<tr>
<td>2</td>
<td>Install the <strong>Network Agent on Linux</strong></td>
</tr>
<tr>
<td>3</td>
<td>Set Up the <strong>Network and App Agents on Linux</strong></td>
</tr>
<tr>
<td>4</td>
<td>Administer the <strong>Network Agent on Linux</strong></td>
</tr>
</tbody>
</table>
Install the Network Agent on Linux

On this page:
- Install the Network Agent Independent of Standalone Machine Agent
- Install the Network Agent as an Extension of the Machine Agent (Bundled)

Related pages:
- Install the Standalone Machine Agent
- ZIP Network Agent Operations
- RPM Network Agent Operations
- DEB Network Agent Operations

Install the Network Agent Independent of Standalone Machine Agent

You can install and run the Network Agent independently of the Standalone Machine Agent using the ZIP, RPM, or DEB installers.

ZIP Network Agent Install

Check System Requirements: See Network Visibility Requirements and Supported Environments.

Download and unzip the agent installer:

1. Download and unzip the Network Agent ZIP package.
2. Enter the following command, where `<network_agent_home>` is the folder where you want to install the agent software:
   ```
   unzip <installer.zip> -d <network_agent_home>
   ```

   The parent path should not include any directory names with spaces:
   ```
   /opt/appdynamics/networkVisibility (OK)
   /opt/appdynamics/network visibility (invalid)
   ```

Install the agent:

1. Log in as root.
2. Go to `<network_agent_home>` and run the install script:
   ```
   cd <network_agent_home>
   ./install.sh
   ```

Important Notes

- If you are using Network Visibility to monitor applications running in Docker containers, you must install the Network Agent in a container as well. See Docker and Network Visibility for additional requirements and recommendations.
- To run the install script, log in as root.
- For all other operations, log in as the designated agent administrator (`<appd-agent-administrator>`) for that host.
4. Start and verify the agent:

   **Login User = Agent Administrator**

   a. Log out and then log in as `appd-agent-administrator`.
   b. Open a CLI window and run the following commands:
      i. `<network-agent-home>/bin/start.sh` *(starts the agent)*
      ii. `<network-agent-home>/bin/appd-netviz.sh status` *(verifies that the agent is running and shows PID of the agent process)*

5. If you experience any problems or issues, see FAQs for Network Visibility.

6. If you are installing the Network Agent as part of a Set Up Network Visibility on Linux workflow, proceed to Set Up the Network and App Agents on Linux.

**RPM Network Agent Install**

> Click here to expand...

**Important Note**

If you are using Network Visibility to monitor applications running in Docker containers, you must install the Network Agent in a container as well. See Docker and Network Visibility for additional requirements and recommendations.

1. Check System Requirements: See Network Visibility Requirements and Supported Environments.
2. If you are logged in to the agent host as a non-root user, log out and then log back in as root.
3. Go to the Downloads page and download the Network Agent installer.
4. Open a terminal window and enter the following commands:
   a. `sudo rpm -ivh <installer.rpm>` *(installs the agent under `opt/appdynamics/netviz`)*
   b. `sudo service appd-netviz start` *(starts the agent)*
   c. `sudo service appd-netviz status` *(verifies that the agent is running)*
5. If you experience any problems or issues, see FAQs for Network Visibility.
6. If you are installing the Network Agent as part of a Set Up Network Visibility on Linux workflow, proceed to Set Up the Network and App Agents on Linux.

**DEB Network Agent Install**

> Click here to expand...

**Important Note**

If you are using Network Visibility to monitor applications running in Docker containers, you must install the Network Agent in a container as well. See Docker and Network Visibility for additional requirements and recommendations.

1. Check System Requirements: See Network Visibility Requirements and Supported Environments.
2. If you are logged in to the agent host as a non-root user, log out and then log back in as root.
3. Go to the Downloads page and download the Network Agent installer.
4. Open a terminal window and enter the following commands:
   a. `sudo dpkg -i <installer.deb>` *(installs the agent under `opt/appdynamics/netviz`)*
      *(starts the agent, outputs "appd-netviz start/running. process <pid #>")*
   b. `sudo service appd-netviz status` *(verifies the agent status)*
5. If the agent does not start, run the following: `sudo service appd-netviz start`
6. If you experience any problems or issues, see FAQs for Network Visibility.
7. If you are installing the Network Agent as part of a Set Up Network Visibility on Linux workflow, proceed to Set Up the Network and App Agents on Linux.

**Install the Network Agent in a Docker Container**

> Click here to expand...
The Network Agent can monitor applications running in Docker containers on the same host. You can deploy one Network Agent per host and collect individual metrics for every local application monitored by a Java App Agent in a container.

AppDynamics recommends that you install the Network Agent in a separate container in Docker Host network mode on the same host as the application containers. This recommendation applies when the App Agent containers are running in Docker Bridge or Docker Host mode.

<table>
<thead>
<tr>
<th>Example 1: App Agents in Separate Network</th>
<th>Example 2: App Agents in Host Network</th>
</tr>
</thead>
</table>

Install the Network Agent and App Agents

1. Paste the sample Dockerfile below into a text file and save the file. This Dockerfile contains commands for installing the Network Agent.
FROM centos:centos7

RUN yum update -y && yum install -y 
    net-tools 
    iproute 
    tcpdump 
    curl 
    unzip 
    sysvinit-tools 
    openssh-clients

WORKDIR /netviz-agent

ARG NETVIZ_ZIP_PKG

# copy NetViz agent contents
ADD $(NETVIZ_ZIP_PKG) .

# run the agent install script
RUN unzip *.zip && ./install.sh 
    && sed -i -e "s|enable_netlib = 1|enable_netlib = 0|g" 
    ./conf/agent_config.lua 
    && sed -i -e "s|WEBSERVICE_IP=.*|WEBSERVICE_IP="0.0.0.0"|g" 
    ./conf/agent_config.lua

# default command to run for the agent
CMD ./bin/appd-netagent -c ./conf -l ./logs -r ./run

2. Navigate to the directory where you saved the Dockerfile. Build the Docker image by running the following command:

docker build --build-arg 
NETVIZ_ZIP_PKG=/path/to/netviz-agent-pkg.zip -t appd-netviz .

3. Push the Docker image to your Docker Trusted Registry.

4. Deploy your Docker image using the sample Docker Compose file below:
version: '2'
services:
  appd-netviz-agent:
    image: path/to/your/docker/image # docker registry image
    network_mode: "host"
    restart: unless-stopped
    ports:
      - '3892:3892'
    cap_add:
      - NET_ADMIN
      - NET_RAW

Alternatively, run the following command:

docker run -d --network=host --cap-add=NET_ADMIN --cap-add=NET_RAW path/to/your/netviz/docker/image

5. Set up the Network Agent. Follow the single-tenant setup instructions on Set Up the Network and App Agents on Linux.
   
   For single-tenant setup, you do not need to enable netlib.

Specify the Docker Host Gateway IP on the App Agent

If you are running the App Agent in a Docker container outside of the Docker Host network (as shown in Example 1 in the diagram above), complete the following steps.

1. In the host for your App Agent, set the following environment variables:

   export APPDYNAMICS_NETVIZ_AGENT_HOST=$(ip -4 addr show docker0 | grep -Po 'inet \K[\d.]+')
   export APPDYNAMICS_NETVIZ_AGENT_PORT=3892

2. Set the values for the port and host address.
   a. If you are using an App Agent version 4.5.1 or earlier:
      Open the `<app-agent-install-dir>/<version-number>/external-services/netviz/netviz-service.properties` file and set the following properties:
      * netviz.agent.host.address
      * netviz.agent.api.service.port
   b. If you are using an App Agent version 4.5.2 or later:
      Pass the exported environment variables to the application containers by running the following command:
Install the Network Agent as an Extension of the Machine Agent (Bundled)

You can install and run the Network Agent as an extension of the Standalone Machine Agent (ZIP only) if you want to install both agents at the same time. This option is available on Linux 32-bit and 64-bit ZIP installers, version 4.4.2 or higher, only. 

Click here to expand...

Important Notes

- If you are using Network Visibility to monitor applications running in Docker containers, you must install the Network Agent in a container as well. See + Docker and Network Visibility for additional requirements and recommendations.
- To run the install script, log in as root.
- For all other operations, log in as the designated agent administrator (<appd-agent-administrator>) for that host.

1. Check System Requirements: See Network Visibility Requirements and Supported Environments.
2. Download and unzip the Standalone Machine Agent:

   Login User = Agent Administrator

   a. Log in to the agent host as <appd-agent-administrator>.
   b. Go to the Downloads page, download the Standalone Machine Agent installer, and unzip it.
   c. Install the Standalone Machine Agent using the Linux ZIP installer as described in the Linux Install Using ZIP with Bundled JRE page.

3. Install the Standalone Machine Agent and Network Visibility extension:

   Login User = root

   a. Log out and then log in as root.
   b. Open a CLI window and enter the following commands:
      i. cd <standalone-machine-agent-home>/extensions/NetVizExtension/agent/
      ii. ./install.sh

4. Enable the Network Visibility extension and start the agent:

   Login User = Agent Administrator

   a. Log out and then log in as <appd-agent-administrator>.
   b. Open the following file in a plain-text editor such as vi or Notepad++:
      <standalone-machine-agent-home>/NetVizExtension/conf/netVizExtensionCon
      f.yml
c. Set the `start` setting to `true` and save the file.
d. Run the following CLI commands:
   i. `<machine-agent-home>/bin/machine-agent -d -p <machine-agent-home>/pidfile` (starts the Machine Agent)
   ii. `<machine-agent-home>/NetVizExtension/agent/bin/appd-netviz.sh` (verifies Network Agent status and PID)

5. If you experience any problems or issues, see FAQs for Network Visibility.
6. If you are installing the Network Agent as part of a Set Up Network Visibility on Linux workflow, proceed to Set Up the Network and App Agents on Linux.
Set Up the Network and App Agents on Linux

On this page:
- Install the Network Agent
- Multi-Tenant Setup (Multiple Apps on the Agent Host)
- Single-Tenant Setup (One App Only on the Agent Host)

Related pages:
- ZIP Network Agent Operations
- RPM Network Agent Operations
- DEB Network Agent Operations
- Install the Java Agent
- Install the Network Agent on Linux

Network Agent version 4.5.7 and Java Agent version 4.5.11 eliminate the need to separately configure the Network Agent in a multi-tenant setup when running multiple nodes on the same machine. If you use the Network Agent version 4.5.7 with Java Agent version 4.5.10 or earlier, you need to configure the Network Agent in a multi-tenant setup. For more details, refer Network Agent and Java Agent Installation Instructions.

Network Agent and Java Agent Installation Instructions

<table>
<thead>
<tr>
<th>If you are using</th>
<th>then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Agent version 4.5.7 or later and Java Agent</td>
<td>Install the Network Agent.</td>
</tr>
<tr>
<td>version 4.5.11 or later</td>
<td></td>
</tr>
<tr>
<td>Network Agent version 4.5.7 or later and Java Agent</td>
<td>Install the Network Agent in a multi-tenant setup or a single tenant</td>
</tr>
<tr>
<td>version 4.5.10 or earlier</td>
<td>setup.</td>
</tr>
<tr>
<td></td>
<td>You do not need to manually enable the netlib parameter during the</td>
</tr>
<tr>
<td></td>
<td>multi-tenant setup, it is enabled by default.</td>
</tr>
<tr>
<td>Network Agent version 4.5.6 or earlier</td>
<td>Install the Network Agent in a multi-tenant setup or a single tenant</td>
</tr>
<tr>
<td></td>
<td>setup.</td>
</tr>
<tr>
<td></td>
<td>You need to manually enable the netlib parameter during the multi-</td>
</tr>
<tr>
<td></td>
<td>tenant setup.</td>
</tr>
<tr>
<td>Java App Agent version 4.4 or earlier</td>
<td>Update the Java Agent.</td>
</tr>
</tbody>
</table>

Install the Network Agent

Click here to expand...

If you are using Java Agent version 4.5.13 or higher, you need not manually enable the socket collection. It is enabled automatically. You can skip the first step and proceed with Step 2.

1. Enable Socket Collection on the App Agent
2. Specify the Docker Host Gateway IP on the App Agent
3. Restart the App (IBM JVM Only)

Enable Socket Collection on the App Agent

1. Configure the Java App Agent startup command to perform the following tasks, as shown in the example:
   - Enable appdynamics.socket.collection.bci.enable for the Network Agent to map network metrics to application flows
     example:
1. Click here to expand...

The following examples illustrate how editing the agent startup command can differ, depending on the framework of the monitored application. For more information, see the relevant section under Agent Installation by Java Framework and the relevant framework documentation.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Example (App Agent ZIP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM WebSphere and InfoSphere</td>
<td>In the WebSphere/InfoSphere UI, navigate to the Java Virtual Machine properties page for the monitored server. Then add javaagent and appdynamics.socket.collection.bci.enable=true as a Generic JVM argument. For example: javaagent:/usr/appd/agents/apm/javaagent.jar -Dappdynamics.socket.collection.bci.enable=true</td>
</tr>
<tr>
<td>Apache Tomcat</td>
<td>Add the full command as a Catalina environment variable in file CATALINA_BASE/bin/setenv.sh (Tomcat 6 and later). For example: export CATALINA_OPTS=&quot;$CATALINA_OPTS -javaagent:/usr/appd/agents/apm/javaagent.jar -Dappdynamics.socket.collection.bci.enable=true&quot;</td>
</tr>
<tr>
<td>Glassfish</td>
<td>Add the full command as a -javaagent JVM option in the GlassFish domain. You can add the agent using the asadmin tool. For example: glassfish4\bin\asadmin create-jvm-options javaagent:/usr/appd/agents/apm/javaagent.jar -Dappdynamics.socket.collection.bci.enable=true*</td>
</tr>
<tr>
<td>JBoss Standalone</td>
<td>Add the full command to the standalone.sh file. For example: export JAVA_OPTS=&quot;$JAVA_OPTS -javaagent:/usr/appd/agents/apm/javaagent.jar -Dappdynamics.socket.collection.bci.enable=true&quot;</td>
</tr>
<tr>
<td>Jetty</td>
<td>Add the full command to the start.ini config file in the Jetty base directory. For example: --exec -javaagent:/usr/appd/agents/apm/javaagent.jar -Dappdynamics.socket.collection.bci.enable=true</td>
</tr>
</tbody>
</table>

2. To verify that the App Agent is running correctly, enter the following command:

```
strings /proc/<app_pid>/environ
```

Specify the Docker Host Gateway IP on the App Agent

These steps are required only if the App Agent is running in a Docker container outside of the Docker Host network.

Error rendering macro 'excerpt-include': No link could be created for '+ Docker and Network Visibility'.

Restart the App (IBM JVM Only)

If the monitored app is running in an IBM JVM, you must restart the app for the Network Agent to detect and monitor any persistent connections.

**Multi-Tenant Setup (Multiple Apps on the Agent Host)**

If you want the Network Agent to monitor multiple app nodes on the same host, you must do the following steps for each App Agent.

Click here to expand...

1. Install or Update the App Agent (if needed)
2. Extract the NetViz External Services folder on the App Agent
3. Preload NetLib and Enable Socket Collection on the App Agent
4. Specify the Docker Host Gateway IP on the App Agent
5 Restart the App (IBM JVM Only)

If you are using Network Agent version 4.5.6 or earlier, enable the netlib parameter manually as described in Step 1.

Enable NetLib on the Network Agent

1. Open the following file in a text editor: `<network_agent_home>/conf/agent_config.lua`
2. Edit the `enable_netlib` parameter as follows:
   ```plaintext
   npm_config = {
     log_destination = "file",
     log_file = "agent.log",
     debug_log_file = "agent-debug.log",
     disable_filter = 1,
     mode = "Advanced",
     enable_netlib = 1,
     lua_scripts_path = ROOT_DIR .. "/scripts/netagent/lua/",
   },
   ```

Install or Update the App Agent (if needed)

The Network Agent requires version 4.4 or higher of the Java App Server Agent. Update each App Agent that is running an earlier version. See Install the Java Agent.

Extract the NetViz External Services folder on the App Agent

1. Navigate to the following folder: `<app-agent-install-dir>/<version-number>/external-services`
2. If not already extracted, extract the `netviz.zip` archive so that the ZIP contents are under the following folder: `<app-agent-install-dir>/<version-number>/external-services`

Preload NetLib and Enable Socket Collection on the App Agent

1. In this step, you must configure the Java App Agent startup command to perform two additional tasks, as shown in the following example:
   - Preload the `appd-netlib` library so the Network Agent can collect network metrics for each individual node (rather than for the entire host)
   - Enable `appdynamics.socket.collection.bci.enable` for the Network Agent to map network metrics to application flows

   ```plaintext
   LD_PRELOAD="/<network_agent_home>/lib/appd-netlib.so \
   java -javaagent:"<app_server_agent_home>/javaagent.jar \
   -Dappdynamics.socket.collection.bci.enable=true
   ```

   The following examples illustrate how editing the agent startup command can differ, depending on the framework of the monitored application. For more information, see the relevant section under Agent Installation by Java Framework and the relevant framework documentation.

   | Framework | Example (App Agent ZIP) |
In the WebSphere/InfoSphere UI, navigate to the **Java Virtual Machine** properties page for the monitored server. Then add `javaagent` and `dappdynamics.socket.collection.bci.enable=true` as a Generic JVM argument. For example:

```
javaagent:/usr/appd/agents/apm/javaagent.jar
-Dappdynamics.socket.collection.bci.enable=true
```

You can add `LD_PRELOAD` as a JVM environment variable. The following example describes how to add an environment variable on a Cognos server using IBM Connections 6.0.0:

1. Log in to the WebSphere Application Server administrative console of the Cognos server.
2. Click Servers > Server Types > WebSphere application servers.
3. Click the cognos_server link.
4. Click JAVA and then Process Management- >Process definition > Environment Entries.
5. Click New to add the following entry:

   ```
   LD_PRELOAD=/usr/appd/agents/netviz/lib/appd-netlib.so
   ```

   Add the full command as a Catalina environment variable in file `CATALINA_BASE/bin/setenv.sh` (Tomcat 6 and later). For example:

   ```
   export CATALINA_OPTS="$CATALINA_OPTS
   LD_PRELOAD=/usr/appd/agents/netviz/lib/appd-netlib.so
   -javaagent:/usr/appd/agents/apm/javaagent.jar
   -Dappdynamics.socket.collection.bci.enable=true"
   ```

   Add the full command as a JVM option in the GlassFish domain. You can add the agent using the `asadmin` tool. For example:

   ```
   glassfish4\bin\asadmin create-jvm-options
   "LD_PRELOAD=/usr/appd/agents/netviz/lib/appd-netlib.so
   -javaagent:/usr/appd/agents/apm/javaagent.jar
   -Dappdynamics.socket.collection.bci.enable=true"
   ```

   Add the full command to the `standalone.sh` file. For example:

   ```
   export JAVA_OPTS="$JAVA_OPTS
   LD_PRELOAD=/usr/appdynamics/agents/netviz/lib/appd-netlib.so
   -javaagent:/usr/appd/agents/apm/javaagent.jar
   -Dappdynamics.socket.collection.bci.enable=true"
   ```

   Add the full command to the `start.ini` config file in the Jetty base directory. For example:

   ```
   --exec
   LD_PRELOAD=/usr/appdynamics/agents/netviz/lib/appd-netlib.so
   -javaagent:/usr/appd/agents/apm/javaagent.jar
   -Dappdynamics.socket.collection.bci.enable=true
   ```

   These steps are required only if the App Agent is running in a Docker container outside of the Docker Host network.

   1. Open the following file: `<app-agent-install-dir>/version-number/external-services/netviz/netviz-service.properties` (You might need to unzip `external-services/netviz.zip` to access this file.)
   2. For the `netviz.agent.host.address`, specify the Gateway IP of the Docker Host network interface. To determine this IP, run the following command on the Docker host:

   ```
   docker inspect <app_container_name> | grep Gateway
   ```
   3. Restart the App Agent.

   To verify that the App Agent is running correctly, enter the following command:

   ```
   strings /proc/<app_pid>/environ
   ```

   Specify the Docker Host Gateway IP on the App Agent.
4. To see if the Docker Host network is connected, run the following command from within the container:

```bash
curl -k -v <host-network-IP>:3892
```

If you receive a response – even if it is 404 Not Found – then the network is connected.

Restart the App (IBM JVM Only)

If the monitored app is running in an IBM JVM, you must restart the app for the Network Agent to detect and monitor any persistent connections.

**Single-Tenant Setup (One App Only on the Agent Host)**

If you want the Network Agent to monitor one app node only on the same host, do the following steps.  

> Click here to expand...

- 1. Install or Update the App Agent (if needed)
- 2. Extract the NetViz External Services folder on the App Agent
- 3. Enable Socket Instrumentation
- 4. Specify the Docker Host Gateway IP on the App Agent
- 5. Restart the App (IBM JVM Only)

Install or Update the App Agent (if needed)

Network Visibility requires a Java App Agent (version 4.4 or higher) installed on the same host as the Network Agent. See Install the Java Agent.

Extract the NetViz External Services folder on the App Agent

1. Navigate to the following folder: `<app-agent-install-dir>/<version-number>/external-services`
2. If not already extracted, extract the netviz.zip archive so that the ZIP contents are under the following folder: `<app-agent-install-dir>/<version-number>/external-services`

Enable Socket Instrumentation

This is required for the agent to map network metrics to application flows.

1. In the Controller, click the gear icon in the top right (⚙️) and choose AppDynamics Agents > App Server Agents.
2. Select the agent in the table and click Configure. The App Server Agent Configuration page appears.
3. Select the application, tier, and node in the treeview (left).
4. Select Use Custom Configuration and then click + (the "plus-sign" button) Create Agent Property.
5. In the Edit Agent Property dialog box, specify:
   - Name = **socket-enabled**
   - Type = **Boolean**
   - Value = **true**
Specify the Docker Host Gateway IP on the App Agent

These steps are required only if the App Agent is running in a Docker container outside of the Docker Host network.

1. Open the following file: <app-agent-install-dir>/<version-number>/external-services/netviz/netviz-service.properties
   (You might need to unzip external-services/netviz.zip to access this file.)
2. For the netviz.agent.host.address, specify the Gateway IP of the Docker Host network interface.
   To determine this IP, run the following command on the Docker host:
   `docker inspect <app_container_name> | grep Gateway`
3. Restart the App Agent.
4. To see if the Docker Host network is connected, run the following command from within the container:
   `curl -k -v <host-network-IP>:3892`
   If you receive a response – even if it is 404 Not Found – then the network is connected.
   Restart the App (IBM JVM Only)

If the monitored app is running in an IBM JVM, you must restart the app for the Network Agent to detect and monitor any persistent connections.
Administer the Network Agent on Linux
ZIP Network Agent Operations

On this page:
- Installing the ZIP Network Agent
- Administering the ZIP Network Agent
- Uninstalling the ZIP Network Agent

Related pages:
- Set Up Network Visibility on Linux
- Multi-Tenant Setup
- Single-Tenant Setup

Installing the Network Agent is generally performed as part of an overall workflow for setting up Network Visibility. The supported workflows are:
- Multi-Tenant Setup
- Single-Tenant Setup

To determine the best workflow for your environment, see Set Up Network Visibility on Linux.

**Installing the ZIP Network Agent**

You can install the agent using the Standalone Machine Agent or the Network Agent ZIP.
- Install using the Standalone Machine Agent ZIP
- Install using the Network Agent ZIP

Install using the Standalone Machine Agent ZIP

You can install the Network Agent as an extension using the Standalone Machine Agent installer. This option is available on Linux 32-bit and 64-bit ZIP installers, version 4.4.2 or higher, only.

1. Check System Requirements: See Network Visibility Requirements and Supported Environments.
2. If you are installing the Network Agent as part of the Multi-Tenant Install workflow, stop each application of interest if it is monitored by an installed App Agent. You will restart each application after you install and set up the Network Agent.
3. Install the Standalone Machine Agent using the Linux ZIP installer as described in the Linux Install Using ZIP with Bundled JRE page.
4. Run the install script using sudo:
   ```bash
cd <standalone-machine-agent-home>/NetVizExtension/agent/
sudo ./install.sh
```
5. Enable the NetVizMonitoring extension:
   a. Open the following file in a plain-text editor such as vi or Notepad++:
      ```bash
      <standalone-machine-agent-home>/NetVizExtension/conf/netVizExtensionConf.yml
      ```
   b. Set the start setting to true and save the file.
6. Start the Standalone Machine Agent. The NetVizMonitoring extension should then start automatically. To start the agent on Linux:
   ```bash
   <machine_agent_home>/bin/machine-agent -d -p <machine_agent_home>/pidfile
   ```
7. Verify that the Network Agent is running:
   ```bash
   <standalone-machine-agent-home>/NetVizExtension/agent/bin/appd-netviz.sh status
   ```
   The output provides the process ID (PID) of the Network Agent process.
   If you experience any problems or issues, see FAQs for Network Visibility.
8. If you are installing the Network Agent as part of a Set Up Network Visibility on Linux workflow, proceed to the next step in the workflow:
   - Multi-Tenant Setup
   - Single-Tenant Setup

Install using the Network Agent ZIP

1. Check System Requirements: See Network Visibility Requirements and Supported Environments.
2. If you are installing the Network Agent as part of the Multi-Tenant Install workflow, stop each application of interest if it is monitored by an installed App Agent. You will restart each application after you install and set up the Network Agent.
3. Verify that you have sudo or root access permissions on the host where you want to install the network agent. (You do not need sudo or root access to run the agent.)
4. Open a command prompt on the host, run the `mount` command, and check the mounted filesystem for the directory where you plan to install the agent. The filesystem should not have the `noexec` and `nsuid` flags set.

5. Download the network-agent package to the machine.

6. In a terminal window, `cd` to the folder where you downloaded the Network Agent ZIP package.

7. Enter the following command, where `<network_agent_home>` is the folder where you want to install the agent software:
   
   ```
   unzip <installer.zip> -d <network_agent_home>
   ```

   The parent path should not include any directory names with spaces:

   ```
   /opt/appdynamics/networkVisibility (OK)
   /opt/appdynamics/network visibility (invalid)
   ```

8. Run the following commands as the administrator or root user on the host machine:
   
   a. The following command is required only if you are logged in as a different user from the one that will configure and run the agent:
      
      ```
      chown -R <network-agent-user-group>:<network-agent-user> <network-agent-home>
      ```

   b. Go to `<network_agent_home>` and run the install script:
      
      ```
      cd <network_agent_home>
      ./install.sh
      ```

9. Start the agent by running the following command as the `<network_agent_user>`:
   
   ```
   <network_agent_home>/bin/start.sh
   ```

10. Verify that the Network Agent is running:

    a. Identify the agent process: `<network-agent-home>/bin/appd-netviz.sh status`
       
       The output provides the process ID (PID) of the Network Agent process.

    b. If you experience any problems or issues, see FAQs for Network Visibility.

11. If you are installing the Network Agent as part of a Set Up Network Visibility on Linux workflow, proceed to the next step in the workflow:

    - Multi-Tenant Setup
    - Single-Tenant Setup

   **Administering the ZIP Network Agent**

   **Run commands as `<network_agent_user>`**

   You should run the following commands as a `<network_agent_user>` with permissions to configure and run the agent on the host machine.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Command</th>
<th>Notes</th>
</tr>
</thead>
</table>
   | Start agent        | `<network-agent-home>/bin/star t.sh` | If the agent starts successfully, you should see the following:  
   |                    |                                |                      Starting AppDynamics Network Monitoring                        |
   |                    |                                |                      appd-netviz...                                              |
   |                    |                                |                      OK                                                      |
   |                    |                                |                      appd-netviz running.. `<pid_of_process>`                          |
   | Stop agent         | `<network-agent-home>/bin/stop.sh` | If the agent stops successfully, you should see the following:  
   |                    |                                |                      Stopping AppDynamics Network Monitoring                        |
   |                    |                                |                      appd-netviz...                                              |
   |                    |                                |                      OK                                                      |
   | Check agent status | `<network-agent-home>/bin/appd-netviz.sh status` | This command shows whether the agent is running or not. You should see the following:  
<p>|                    |                                |                      appd-netviz start/running, process <code>&lt;pid #&gt;</code>                    |</p>
<table>
<thead>
<tr>
<th>Restart agent</th>
<th><code>network-agent-home</code>/bin/appd-netviz.sh restart</th>
<th>If the agent restarts successfully, you should see the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Stopping AppDynamics Network Monitoring appd-netviz...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Starting AppDynamics Network Monitoring appd-netviz...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>appd-netviz running.. <code>&lt;pid_of_process&gt;</code></td>
</tr>
</tbody>
</table>

**Uninstalling the ZIP Network Agent**

1. Open a terminal and `cd` to the `<network-agent-home>` directory.
2. Stop the agent by running the command: `./bin/stop.sh`
3. `cd` to the `<network-agent-home>` parent directory: `cd ..`
4. Remove the agent directory by running the following command: `rm -rf <network-agent-home>`
RPM Network Agent Operations

On this page:

- Installing the RPM Agent
- Administering the RPM Agent
- Uninstalling the RPM Agent

Related pages:

- Set Up Network Visibility on Linux
- Multi-Tenant Install
- Single-Tenant Install with App Restart
- Single-Tenant Install with no App Restart

Installing the Network Agent is generally performed as part of an overall workflow for setting up Network Visibility. The three supported workflows are:

- Multi-Tenant Install
- Single-Tenant Install with App Restart
- Single-Tenant Install with no App Restart

To determine the best workflow for your environment, see Set Up Network Visibility on Linux.

**Installing the RPM Agent**

1. Check System Requirements: See Network Visibility Requirements and Supported Environments.
2. If you are installing the Network Agent as part of the Multi-Tenant Install workflow, stop each application of interest if it is monitored by an installed App Agent. You will restart each application after you install and set up the Network Agent.
3. Verify that you have `sudo` or `root` access permissions on the machine where you want to install the network agent. (You do not need `sudo` or `root` access to run the agent.)
4. Open a command prompt on the host, run the `mount` command, and check the mounted filesystem for the directory where you plan to install the agent. The filesystem should not have the `noexec` and `nsuid` flags set.
5. Download the network-agent package to the machine.
6. Verify that you have `sudo` or `root` access permissions on the machine where you want to install the agent.
7. Download the agent package to the machine.
8. Open a terminal window and enter the following command: `sudo rpm -ivh <installer.rpm>`
   The network agent software is installed under `/opt/appdynamics/netviz`.
9. To start the agent, enter the following command: `sudo service appd-netviz start`
10. Verify that the Network Agent is running: `sudo service appd-netviz status`
11. If you experience any problems or issues, see FAQs and Troubleshooting for Network Visibility.
12. If you are installing the Network Agent as part of a Set Up Network Visibility on Linux workflow, proceed to the next step in the workflow:
   - Multi-Tenant Install
   - Single-Tenant Install with App Restart
   - Single-Tenant Install with no App Restart

**Administering the RPM Agent**

You should run the following commands as a `<network_agent_user>` with permissions to configure and run the agent on the host machine.
<table>
<thead>
<tr>
<th>Operation</th>
<th>Command</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start agent</td>
<td>`sudo service</td>
<td>If the agent starts successfully, you should see the following:</td>
</tr>
<tr>
<td></td>
<td>appd-netviz start</td>
<td>Starting appd-netviz (via systemctl): [ OK ]</td>
</tr>
<tr>
<td>Stop agent</td>
<td>`sudo service</td>
<td>If the agent stops successfully, you should see the following:</td>
</tr>
<tr>
<td></td>
<td>appd-netviz stop</td>
<td>Stopping appd-netviz (via systemctl): [ OK ]</td>
</tr>
<tr>
<td>Restart agent</td>
<td>`sudo service</td>
<td>If the agent restarts successfully, you should see the following:</td>
</tr>
<tr>
<td></td>
<td>appd-netviz restart</td>
<td>Stopping appd-netviz (via systemctl): [ OK ]</td>
</tr>
<tr>
<td>Check agent</td>
<td>`sudo service</td>
<td>The following example illustrates status output:</td>
</tr>
<tr>
<td>status</td>
<td>appd-netviz status</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>appd-netviz.service - Appdynamics Network Monitoring</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Loaded: loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(/usr/lib/systemd/system/appd-netviz.service; disabled; vendor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active: active (running) since Thu 2016-09-29 14:52:42 PDT; 14s ago</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Process: 3981</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ExecStart=/opt/appdynamics/netviz/bin/appd-netmon (code=exited);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Process: 3973 ExecStartPre=/bin/chown -R appd-netviz:/opt/appdynamics/netviz /opt/appdynamics/netviz/run/ (code=ex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Process: 3968 ExecStartPre=/bin/mkdir -p /opt/appdynamics/netviz/run/ (code=ex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Main PID: 3983 (appd-netmon)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tasks: 12 (limit: 512)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CGroup: /system.slice/appd-netviz.service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3983 /opt/appdynamics/netviz/bin/appd-netmon</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3984 appd-netagent -l</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/opt/appdynamics/netviz/scripts/netagent/lua/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3985 appd-netcor -l</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/opt/appdynamics/netviz/scripts/netcor/lua/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sep 29 14:52:42 localhost.localdomain systemd[1]:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Starting Appdynamics Network M</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sep 29 14:52:42 localhost.localdomain systemd[1]:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Started Appdynamics Network Mo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lines 1-15/15 (END)</td>
</tr>
</tbody>
</table>

**Uninstalling the RPM Agent**

To uninstall the Network Agent, run the following command:

```
sudo rpm -e appd-netviz
```

This command generates no output. To confirm that the agent is removed, run the status command again: `sudo service appd-netviz status`
DEB Network Agent Operations

On this page:
- Installing the DEB Network Agent
- Administering the DEB Network Agent
- Uninstalling the DEB Network Agent

Related pages:
- Set Up Network Visibility on Linux
- Multi-Tenant Install
- Single-Tenant Install with App Restart
- Single-Tenant Install with no App Restart

Installing the Network Agent is generally performed as part of an overall workflow for setting up Network Visibility. The three supported workflows are:
- Multi-Tenant Install
- Single-Tenant Install with App Restart
- Single-Tenant Install with no App Restart

To determine the best workflow for your environment, see Set Up Network Visibility on Linux.

Installing the DEB Network Agent

1. Check System Requirements: See Network Visibility Requirements and Supported Environments.
2. If you are installing the Network Agent as part of the Multi-Tenant Install workflow, stop each application of interest if it is monitored by an installed App Agent. You will restart each application after you install and set up the Network Agent.
3. Verify that you have sudo or root access permissions on the machine where you want to install the agent. (You do not need sudo or root access to run the agent.)
4. Open a command prompt on the host, run the mount command, and check the mounted filesystem for the directory where you plan to install the agent. The filesystem should not have the noexec and nsuid flags set.
5. Download the agent package to the machine.
6. Enter the following command:
   ```bash
   sudo dpkg -i <installer.deb>
   ```
   The network agent software is installed under /opt/appdynamics/netviz.
   The agent should start automatically with the following message: appd-netviz start/running. process <pid #>

   To verify the Network Agent status, run the following: `sudo service appd-netviz status`

7. If the agent does not start, run the following: `sudo service appd-netviz start`
8. If you experience any problems or issues, see FAQs and Troubleshooting for Network Visibility.
9. If you are installing the Network Agent as part of a Set Up Network Visibility on Linux workflow, proceed to the next step in the workflow:
   - Multi-Tenant Install
   - Single-Tenant Install with App Restart
   - Single-Tenant Install with no App Restart

Administering the DEB Network Agent

<table>
<thead>
<tr>
<th>Operation</th>
<th>Command</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start agent</td>
<td><code>sudo service appd-netviz start</code></td>
<td>If the agent starts successfully, you should see the following: appd-netviz start/running. process &lt;pid #&gt;</td>
</tr>
<tr>
<td>Stop agent</td>
<td><code>sudo service appd-netviz stop</code></td>
<td>If the agent stops successfully, you should see the following: appd-netviz stop appd-netviz stop/waiting</td>
</tr>
</tbody>
</table>
### Restart agent

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sudo service appd-netviz restart</code></td>
</tr>
</tbody>
</table>

### Check Agent Status

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| `sudo service appd-netviz status` | This command shows whether the agent is running or not. You should see the following:  
appd-netviz start/running, process <pid #> |

### Uninstalling the DEB Network Agent

The network agent software is installed under `/opt/appdynamics/netviz`. To uninstall the agent, run the following command.

```
sudo dpkg -P appd-netviz
```
Managing Network Agents in the Controller

On this page:
- Controller Operations for Network Agents
- Checking Network Agent Status

Related Pages:
- Tiers and Nodes
- FAQs for Network Visibility
- ZIP Network Agent Operations
- DEB Network Agent Operations
- RPM Network Agent Operations

Controller Operations for Network Agents

Using an AppDynamics Administrator account, click Settings > AppDynamics Agents > Network Visibility Agents to see details and manage your installed Machine Agents. You can perform most operations on multiple agents: shift- or ctrl-click the agents of interest, right-click, and choose the operation. These agent settings are defined in the Controller and retain their state when an agent is stopped and restarted.

Right-click on one or more agents to do the following:
- View Packet Capture Configuration
- Start Packet Capture
- Disable Agent(s) or Enable Agent(s)
- Change Dynamic Monitoring Mode
- Delete Agent from System

View Packet Capture Configuration

View and edit configuration settings for a single network agent. See Packet Captures.

Start Packet Capture

Start a packet capture on one or more agents. See Packet Captures.

Disable Agent(s) or Enable Agent(s)

When disabled, the agent temporarily suspends its collection of network metrics. You can quickly re-enable the agent when you want.

While disabled, the agent continues to consume a Network Visibility license. The agent maintains a "heartbeat" connection to the Controller so you can enable it again quickly. The agent persists its enabled/disabled state after it restarts and resumes all normal activities after it is re-enabled.

Change Dynamic Monitoring Mode

Defines the set of metrics that the agent collects. In general, you should run the agent in KPI unless you need to diagnose network performance on the node or Connections monitored by that agent. See Dynamic Monitoring Mode and Network Visibility.

Delete Agent from System

The purpose of the delete functionality on the admin page is to delete the underlying Network Visibility metadata for an orphaned network agent. A network agent is considered "orphaned" when the agent host no longer has any App Server agents installed or running. To delete a network agent, you must first delete all APM nodes on that host. Account Administrator privileges are required to delete network agents.

Checking Network Agent Status

To check the status of your Network Agents, go to Tiers & Nodes and select Network in the Show Data pull-down menu.
Network Visibility with Kubernetes

You can use Network Visibility to monitor applications running on Kubernetes. Network Visibility isolates an application's network issues from its application issues. It monitors an application's network interactions and reports key performance metrics in the context of application performance monitoring.

How It Works

To monitor the communication between pods and between nodes, the agent opens up a TCP port in each node for app containers to communicate with the Network DaemonSet container using a REST API. The agent is deployed as a DaemonSet in each node that has host mode enabled. Follow the steps on this page to create a Docker image for the DaemonSet and configure the agent.

Before You Begin

- Ensure that you have at least one pod with a Java Agent (version 4.4 or higher) deployed to the same cluster as the Network Agent.
- Ensure that TCP port 3892 is not already used by the node. Port 3892 will be used by the application pods to communicate with the DaemonSet.

Creating a Docker Image

To deploy Network Visibility with Kubernetes, you must first create a Docker image for the Network Visibility DaemonSet and push the image to your Docker Trusted Registry.

1. Paste the sample Dockerfile below into a text file and save the file.
FROM centos:centos7
RUN yum update -y && yum install -y 
   net-tools \
   tcpdump \
   curl \
   unzip \
   sysvinit-tools \
   openssh-clients
WORKDIR /netviz-agent
ARG NETVIZ_ZIP_PKG
# copy NetViz agent contents
ADD ${NETVIZ_ZIP_PKG} .

# run the agent install script
RUN unzip *.zip && ./install.sh \
    && sed -i -e "s|enable_netlib = 1|enable_netlib = 0|g" 
    ./conf/agent_config.lua \
    && sed -i -e "s|WEBSERVICE_IP=.*|WEBSERVICE_IP="0.0.0.0"|g" 
    ./conf/agent_config.lua
# default command to run for the agent
CMD ./.bin/appd-netagent -c ./conf -l ./logs -r ./run

2. Navigate to the directory where you saved the Dockerfile. Build the Docker image by running the following command:

```
$ docker build --build-arg
   NETVIZ_ZIP_PKG=/path/to/netviz-agent-pkg.zip -t appd-netviz .
```

3. Push the Docker image to your Docker Trusted Registry.

**Configuring Network Visibility with Kubernetes**

1. Paste the configuration below into a yaml file. This configuration file is used for deploying the Network Visibility agent.
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: DaemonSet
metadata:
  name: appd-netviz-agent
spec:
  selector:
    matchLabels:
      name: appd-netviz-agent
  template:
    metadata:
      name: appd-netviz-agent
      labels:
        name: appd-netviz-agent
    spec:
      hostIPC: true
      hostNetwork: true
      containers:
      - name: appd-netviz-agent
        image: path/to/your/Docker/image # docker registry image
        resources:
          requests:
            memory: "250Mi"
            cpu: "0.5"
          limits:
            memory: "2Gi"
            cpu: "1"
        securityContext:
          capabilities:
            add: ["NET_ADMIN", "NET_RAW"]
        ports:
        - containerPort: 3892
          hostPort: 3892
        imagePullSecrets:
        - name: your-registry-key # add the registry key, kubectl create secret ...

2. In the configuration file, update these fields:
   - image (under containers): The file path to the DaemonSet image in your Docker Trusted Registry.
   - name (under imagePullSecrets): The key for your Docker Trusted Registry.

3. Deploy the Network Visibility agent for Kubernetes by running the following command:
   $ kubectl apply -f MyConfigFile.yaml

Configure Network Visibility to Monitor Application Pods

After installing Network Visibility for Kubernetes, you'll need to correlate Network Visibility with a Java agent. This allows you to map network metrics to application flows. To do this, deploy at least one pod with a Java Agent (version 4.4 or higher) to the same cluster as the Network Agent.
1. Open the application’s deployment configuration yaml file with Kubernetes in a text editor and set the
APPDYNAMICS_NETVIZ_AGENT_HOST and APPDYNAMICS_NETVIZ_AGENT_PORT values, as shown:

```yaml
- name: APPDYNAMICS_NETVIZ_AGENT_HOST
  valueFrom:
    fieldRef:
      fieldPath: status.hostIP
- name: APPDYNAMICS_NETVIZ_AGENT_PORT
  value: 3892
```

If you are using an app agent that is 4.5.2 or older, you must open the `<app-agent-install-dir>/<version-number>/external-services/netviz/netviz-service.properties` file to set these values.

2. In the Controller UI, enable socket instrumentation so that you can map network metrics to application flows. For instructions on enabling socket instrumentation, see Set Up the Network and App Agents.
## Set up Network Visibility on Windows

The following table outlines the required steps for setting up Network Visibility on Windows:

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Review the Network Visibility Requirements and Supported Environments</td>
</tr>
<tr>
<td>2</td>
<td>Install the Network Agent on Windows</td>
</tr>
<tr>
<td>3</td>
<td>Install the App Agents on Windows</td>
</tr>
</tbody>
</table>
|      | **Note**  
Network Visibility on Windows supports only Java applications. |
| 4    | Administer the Network Agent on Windows |
| 5    | Verify the Network Visibility Installation on Windows |
Install the Network Agent on Windows

On this page:
- Agent Install Directory
- Network Agent and Java Agent Installation Instructions
- Install the Network Agent on Windows using the ZIP Installer
- Uninstall Network Agent Using the ZIP

Related pages:
- License Management
- Java Supported Environments
- Set Up Network Visibility on Linux

The Network Visibility on Windows requires a supported version of the Network Agent to be installed on the server. To monitor Java applications in your Controller, install the AppDynamics Java Agent on each server that hosts applications that you want to monitor. You need a Network Visibility license to enable the Network Agent and its features. No additional configuration is required on your AppDynamics Controller, as Network Visibility is enabled by default.

Agent Install Directory

Network Agent is packaged as a zip bundle. The Network Agent install directory \<network_agent_home> is the directory where you install the agent. You can unzip the bundle at any location which is your \<network_agent_home> directory.

It contains the following administrative files:
- install.bat to install and start the Network Visibility Agent
- start.bat to start the Network Visibility Agent
- status.bat to check the status of the Network Visibility Agent
- stop.bat to stop the Network Visibility Agent
- uninstall.bat to uninstall the Network Visibility Agent

As part of the installation, install.bat installs the required VC redistributable on the system along with NPcap, if it is not present already. If NPcap is already present on the system, then installation of NPcap is skipped. To listen to the traffic on localhost, NPcap should be running in LoopBack mode. If NPcap is already installed on the system and not running in loopback mode, ensure that you re-install it in loopback mode.

Network Agent and Java Agent Installation Instructions

Depending on the Network Agent version and Java Agent version, you can install the Network Agent in a multi-tenant or single-tenant setup. Refer to the following table for more information.

<table>
<thead>
<tr>
<th>If you are using</th>
<th>Multi-tenant Setup</th>
<th>Single-tenant Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Agent version 4.5.7 or later and Java Agent version 4.5.11 or later</td>
<td>Install the Network Agent on windows</td>
<td>Install the Network Agent on windows</td>
</tr>
<tr>
<td>Network Agent version 4.5.7 or later and Java Agent version 4.5.10 or earlier</td>
<td>Not supported</td>
<td>Install the Network Agent on windows</td>
</tr>
</tbody>
</table>

Install the Network Agent on Windows using the ZIP Installer

The following steps describe how to install and run the Network Agent for Windows using the ZIP installer.

1. Review the Network Visibility Requirements and Supported Environments.
2. Download the Network Agent Windows ZIP package from downloads portal and unzip the package.
   a. Verify your administrator permissions on the host where you want to install the Network Agent.
   b. Extract the contents to the agent installation directory \<network_agent_home>.
3. Double-click the install.bat file to install the Network Agent and start the AppDynamics Network Visibility Service.
4. To verify if the Network Agent is running, navigate to the folder that contains the `status.bat` file.
   a. Double-click the `status.bat` file.
   b. Locate the **AppDynamics Network Visibility Service** and verify the status in the **Status** column. 
      Alternately, check if AppDynamics NetViz Service is listed in `services.msc`.

For more information on various commands, see **Administer the Network Agent on Windows**.

### Uninstall Network Agent Using the ZIP

To uninstall the Network Agent:

1. Open a Windows command prompt and `cd` to the `<network-agent-home>` directory.
2. Uninstall the Network Agent by running the command: `\<network-agent-home>\uninstall.bat`. 

   If you run the `install.bat` script without administrator privileges, a dialog box is displayed: "Do you want to allow this app to make changes to your device?"

   Click Yes to continue with the installation.
Install the App Agent on Windows

On this page:
- Install the Java App Agent on Windows
- Set Up the Java App Agent on Windows

Related pages:
- Java Agent
- Install the Java Agent

Instrumenting an application adds the AppDynamics Application Agent, known as an App Agent into the runtime process of the application. An App Server is a server application that is constantly running. You can deploy applications on the App Server.

Install the Java App Agent on Windows

To monitor Java applications in the Controller, you need to install the AppDynamics Java Agent, version 4.5.4 or later, on each server that hosts applications that you want to monitor.

- To install using the Agent Download Wizard in the Controller, see Java Agent.
- To install and configure the AppDynamics Java Agent into the application JVM, see Install the Java Agent.

Set Up the Java App Agent on Windows

If you want the Network Agent to monitor an app node on the same host:

1. Enable Socket Instrumentation
2. Restart the App (IBM JVM Only)

Enable Socket Instrumentation

These steps are required for the agent to map network metrics to application flows.

1. In the Controller, click the gear icon in the top right (⚙️) and choose AppDynamics Agents > App Server Agents.
2. Select the agent in the table and click Configure. The App Server Agent Configuration page appears.
3. Select the application, tier, and node in the tree-view (left).
4. Select Use Custom Configuration and then click + (the "plus-sign" button). The Create Agent Property dialog box appears.
5. In the Create Agent Property dialog box, specify:
   a. Name = socket-enabled
   b. Type = Boolean
   c. Value = true
Restart the App (IBM JVM Only)

If the monitored app is running in an IBM JVM, you must restart the app for the Network Agent to detect and monitor any persistent connections.
Administer the Network Agent on Windows

Depending on how you installed and deployed the Network Agent, you may have more than one method to administer the Network Agent. AppDynamics provides both GUI and command-line tool for common operations, such as starting and stopping Network Agents and their services. See Managing Network Agents in the Controller.

This section describes how to administer the Network Agent using command-line tools for the Network Agent ZIP. The Network Agent can be managed using the following commands:

- install
- start
- status
- stop
- uninstall

Ensure that you run all the scripts using administrator privilege. If you run the scripts without administrator privilege, the scripts prompt for the privilege request after a 3 second timeout. You can start these scripts from Windows Explorer or from the command line. If you run the scripts from the command line, the install, start and stop scripts have a -quiet option. The -quiet option runs the scripts without prompts, messages, or dialog boxes.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Command</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install agent</td>
<td><code>&lt;network-agent-home&gt;\install.bat</code></td>
<td>The install.bat installs the Network Visibility agent and starts the AppDynamics Network Visibility Service.</td>
</tr>
<tr>
<td>Start agent</td>
<td><code>&lt;network-agent-home&gt;\start.bat</code></td>
<td>If the agent starts successfully, the following message is displayed: Starting the network agent service  SERVICE_NAME: NetVizService  TYPE : 10 WIN32_OWN_PROCESS  STATE : 4 RUNNING  (STOPPABLE, NOT_PAUSABLE, ACCEPTS_PRESHUTDOWN)  WIN32_EXIT_CODE : 0 (0x0)  SERVICE_EXIT_CODE : 0 (0x0)  CHECKPOINT : 0x0  WAIT_HINT : 0x0  PID : 7556  FLAGS :</td>
</tr>
<tr>
<td>Check agent status</td>
<td><code>&lt;network-agent-home&gt;\status.bat</code></td>
<td>This command shows if the agent is running or not.  SERVICE_NAME: NetVizService  TYPE : 10 WIN32_OWN_PROCESS  STATE : 4 RUNNING  (STOPPABLE, NOT_PAUSABLE, ACCEPTS_PRESHUTDOWN)  WIN32_EXIT_CODE : 0 (0x0)  SERVICE_EXIT_CODE : 0 (0x0)  CHECKPOINT : 0x0  WAIT_HINT : 0x0</td>
</tr>
</tbody>
</table>
### Managing Network Agent on Windows

You can start or stop the Network Agent using `start.bat` or `stop.bat`. You require admin privileges to start or stop the Network Agent.

You can view the services that are currently running on the system using AppDynamics NetViz Service.

| Stop agent | \<network-agent-home\>\stop.bat | If the agent stops successfully, the following message is displayed:
|---|---|---
| | | SERVICE_NAME: NetVizService
| | | TYPE : 10 WIN32_OWN_PROCESS
| | | STATE : 1 STOPPED
| | | WIN32_EXIT_CODE : 0 (0x0)
| | | SERVICE_EXIT_CODE : 0 (0x0)
| | | CHECKPOINT : 0x0
| | | WAIT_HINT : 0x0

| Uninstall agent | \<network-agent-home\>\uninstall.bat | If the agent uninstalls successfully, the following message is displayed:
|---|---|---
| | | Stopping service NetVizService.
| | | NetVizService is stopped.
| | | Service AppDynamics NetViz Service uninstalled.
Network Visibility Metrics

Related pages:
- KPI Metrics in Network Dashboard and Application Flow Map
- KPI Metrics in Right-Click Dashboards
- TCP Connection Metrics in Metric Browser
- Node Metrics in Network Dashboard and Metric Browser

Network Visibility can collect the following types of metric:

- **KPI Metrics in Network Browser**
  Key Performance Indicator (KPI) metrics are essential measures of whether a network element is performing successfully. The Network Browser shows KPIs for tiers, nodes, network links, and the entire application.

- **KPI Metrics in Right-Click Dashboards**
  You can right-click on a network element and perform network troubleshooting in the right-click dashboard. These dashboards make it easy to drill down to network elements and metrics that correlate with application-performance issues.

- **TCP Connection Metrics**
  When you narrow down a network problem to a specific network location, you can configure Network Agents to collect advanced metrics for the relevant connections.

- **Node Metrics in Network Dashboard and Metric Browser**
  The Network Agent also collects node-level metrics for monitoring the TCP stack health, collisions, and errors on physical interfaces.
KPI Metrics in Network Dashboard and Application Flow Map

On this page:
- Viewing Network KPIs
- Throughput
- Latency
- TCP Loss (mille)
- Errors
- PIE (Performance Impacting Events)

Related pages:
- KPI Metrics in Right-Click Dashboards
- TCP Connection Metrics in Metric Browser
- Node Metrics in Network Dashboard and Metric Browser
- Dynamic Monitoring Mode and Network Visibility

Some network metrics are considered Key Performance Indicators (KPIs) because they are essential measures of whether the network or a specific part of the network, is performing successfully. Network Agents collect network KPIs for all monitored tiers, nodes, network links, and application flows. You can view KPI metrics for the following elements:

**Viewing Network KPIs**

For the entire Application

The *Network Dashboard* includes Throughput, TCP Loss, and PIE charts that measure network health for the entire application.

For a Network Link

- **Network Dashboard** – The flow map shows KPIs for each link. Click on a link to view KPI time charts and KPIs per Connection.
- **Right-click dashboard** – Right-click on a link in the Network Dashboard and choose View Metrics. See KPI Metrics in Right-Click Dashboards.

For a Tier

- **Network Flow Map** – Click on a tier to view KPI time charts.
- **Right-click dashboard** – Right-click on a tier in the Network Dashboard and choose View Metrics. See KPI Metrics in Right-Click Dashboards.

For a Node

- **Tiers & Nodes page** – Set the Show Data menu to Network. The table shows network KPIs and the Network Agent status for each node.
- **Node Dashboard** – Right-click on the node and choose View Metrics. See KPI Metrics in Right-Click Dashboards.

For a Connection

Network agents do not collect metrics for Connections by default. To collect metrics for the Connections used by a specific node, change the Monitoring Mode on the agent. See Dynamic Monitoring Mode and Network Visibility.

**Connections Explorer** – Go to the Network Dashboard and click Connections. The Explorer shows network KPIs for all network links. Drill down into a network link to see the Connections used on that link. If a Network Agent has collected metrics for a Connection, you can

- View network KPIs in the Explorer table.
- Right-click and choose View Metrics. See KPI Metrics in Right-Click Dashboards.
- Right-click and choose View Metric Browser. See TCP Connection Metrics in Metric Browser.

**KPI Metric Descriptions**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
</table>

Copyright © AppDynamics 2012-2019
| **Throughput** | The average rate of Application X traffic sent and received over one Connection or a set of Connections.  

- Correlated spikes along with Response Time indicate bottlenecks. | **KPI** |
| **Latency** | The average TCP Latency (round-trip time) for a packet to go from one application tier the other and back again. When TCP effects are not significant, Latency primarily measures network round trip times. Network latency is primarily a factor of the physical distance between two nodes and remains consistent over time.  

- Correlated spikes along with Response Time indicate that TCP effects between two tiers (such as packet queueing and delayed acknowledgments) are affecting application performance. | **KPI** |
| **TCP Loss (milie)** | The average number of packets (per 1000) that were sent but never received. Packet loss can degrade application performance significantly. TCP detects and retransmits lost packets; this ensures reliable transmission but introduces delays.  

- Correlated spikes with Response Time indicates that network congestion between the sender and receiver is affecting application performance.  
- Correlated spikes with Throughput indicates that a tier is sending more data than the receiving tier or the network can handle. | **KPI** |
| **Errors** | The number of TCP messages sent indicating an error in setting up the connection (SYN errors) or tearing down the connection (FIN errors) for a specific TCP session. | **KPI** |
| **PIE (Performance Impacting Events)** | The rate of PIE (Performance Impacting Events), which indicate performance issues on one node, both nodes, and/or the network between two nodes:  

- **Client Limited** and **Client Zero Window** events indicate that the Client node is receiving data at a higher rate than it can process.  
- **TCP Retransmission Timeouts (RTO)** events indicate that the network is losing packets between the client and the server node. RTOs can cause severe performance degradations.  
- **Server Limited** and **Server Zero Window** events indicate that the Server node is receiving data at a higher rate than it can process. | **KPI** |
KPI Metrics in Right-Click Dashboards

To diagnose a network element, right-click and choose View Metrics. You can troubleshoot the following network elements:

- Tier – The performance chart (top left) shows the rate or application performance outliers on the relevant nodes (Errors and Slow/Very Slow/Stalled Calls) and the Key Performance Indicators for all Connections used by those nodes (Errors and PIE).
- Network Link – The performance chart (top left) shows the rate of Performance Impacting Events for all member Connections of that link.
- Network Connection – You can troubleshoot a Connection from the Connection Explorer or from the Connections tab in a link popup.
- You can troubleshoot a transaction from a Transaction Snapshot.
  - You can right-click on an application flow and choose View Network Metrics
  - You can also drill down to a node where a transaction delay/stall/error occurred and go to the Network tab.

To troubleshoot a tier, link, or connection in the Network Browser, right-click the network element and choose View Metrics. The top-left chart in the dashboard shows the overall network and/or network/application performance of the element. To troubleshoot the element, look for correlations between the performance chart (top left) and the other charts on the page.

Tip
You can switch between the linear and logarithmic scale in each chart to best highlight metric spikes and variations. Click on the settings button (top-right corner of the chart) to switch between scales.
Right-Click Chart Descriptions

<table>
<thead>
<tr>
<th>Chart</th>
<th>Description</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
</table>

Network Element Performance Chart

Root-Cause Correlation Charts
<table>
<thead>
<tr>
<th>Network Impact on Transactions (Tiers)</th>
<th>This chart highlights the possible impact of Performance Impacting Events on the app-transaction outliers (Transaction Errors and Slow, Very Slow, and Stalled Calls shown in the Transaction Scorecard). If you see spikes in transaction outliers and correlated spikes in PIE or errors, this indicates that the network is affecting application performance. Look for correlated spikes in the other charts to identify specific issues and root causes: connection errors, packet loss, retransmissions, high-latency connections, and so on.</th>
<th>KPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Stack KPIs</td>
<td>Interface errors indicate issues on the node's physical interfaces. See Interface Metrics. TCP Wait sockets can result in significant delays and/or errors for the application or service that relies on that socket. A lot of simultaneous WAIT sockets can prevent applications and services from creating new connections.</td>
<td>KPI</td>
</tr>
</tbody>
</table>
| Network PIE - Contributors           | Performance Impacting Elements (PIE) are useful for identifying the location of actual or potential bottlenecks:  
  - Client Limited and Client Zero Window events indicate a possible problem on the client node.  
  - RTOs indicate a possible problem on the network path between two nodes.  
  - Server Limited and Server Zero Window events indicate a possible problem on the server node. | KPI |
| Network Errors - Contributors        | This chart is useful for identifying network errors that can affect application performance:  
  - FIN Errors – Errors when tearing down a TCP connection.  
  - Syn Blackholes - Connection attempts that went unanswered and resulted in a failure  
  - Syn Resets – Connection attempts that were explicitly refused by the other host  
  - RST on Established - TCP connection resets | KPI |
| Throughput                           | Traffic throughput for the application of interest on the network element. This chart measures application payload data only; TCP, IP, and other packet headers are excluded. | KPI |
| Connection Rate Info                 | Spikes in this chart that coincide with spikes in slows calls or applications errors indicate that there might be an issue with how the application is using TCP.  
  - New Connections – Number of new connection attempts  
  - Connection Errors – The rate of TCP connection setup (SYN) errors and connection teardown (FIN) errors  
  - RSTs (resets) – A TCP Reset is an immediate closing of a connection. Not all connection resets indicate a problem, but it is good practice to investigate any spike in resets that coincides with a spike in application errors or slow transactions/calls. Connection resets can occur for various reasons, such as:  
    - Inability to create connections by the Server.  
    - Intermediate network elements such as Load Balancers, Firewalls, etc. due to misconfiguration or other errors.  
    - Current Established Connections | KPI |
| TCP Loss (mille)                     | The number of packets lost (sent but not received) per 1000 packets sent. "Per mille" is a percentage with one additional digit of precision. TCP detects lost packets and retransmits each lost packet until it receives an ACK (acknowledgment) from the peer. Spikes in TCP Loss generally indicate that the network is overutilized. | KPI |
| **Retransmissions per Minute** | - Data Retransmits – The percentage of packets that were retransmitted. This metric includes SYN and FIN retransmissions.  
- SACK Retransmits – The percentage of data packets that were retransmitted due to selective acknowledgments (SACK, a TCP Feature). | **KPI** |
| **Latency (RTT) Comparison** | This chart compares the average TCP round-trip times (RTTs) for different types of packet request/responses.  
- Handshake RTT – Average RTT for the initial 3-way handshake (SYN, SYN/ACK, ACK) to set up a connection. If Handshake RTTs are significantly higher than Initial RTTs, this indicates that the client node is taking a long time to set up the connection.  
- Initial RTT – Average RTT for the initial SYN packets (between SYN and SYN-ACK or SYN-ACK and ACK). If Initial RTTs are high, the delay might be due to an intervening firewall, asymmetric routing, or other network issues that causes the network to treat SYN packets differently from data packets.  
- RTT – Average TCP RTT for data request/responses—that is, the time for TCP Data segments to be transmitted and acknowledged from the peer. This is different from Application Response Time metrics that measure the time an application takes to process a request. | **KPI** |
| **Connection Lifetime** | TCP is most efficient when long, stable connections are used. Connection setups and tear downs are very time-consuming and resource-intensive. The more short-term connections get generated, the worse TCP performance will be because most of the time is spent in creating connections and because of "Slow-Start" in TCP optimal TCP bandwidth is not achieved. | **KPI** |
The network agent also collects various TCP socket metrics at the host level for the local node on which it is installed. These metrics are useful for monitoring the TCP stack health, collisions, and errors on all the physical interfaces, and CPU consumption of the network agent.

**TCP Socket Metrics in the Network Dashboard**

These metrics measure the overall TCP socket health of an individual node. To view these metrics, select a node: go to the Network Dashboard, click on a tier, go to the Nodes tab, and select the node of interest. The TCP Socket metrics for the selected node appear on the right side of the dashboard.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Established</td>
<td>The number of sockets that are in the established state.</td>
<td>KPI</td>
</tr>
<tr>
<td>Embryonic</td>
<td>The number of sockets that are in the process of being set up.</td>
<td>KPI</td>
</tr>
</tbody>
</table>
Wait
The number of sockets that are unavailable for new connections because they are in the process of being closed. When a socket sends a Connection Close (FIN) message, the socket goes through three states:

1. FIN_WAIT1 – Socket has sent the FIN and is waiting for an ACK from the remote socket.
2. FIN_WAIT2 – Socket receives ACK, waits for FIN (final connection close) from the remote socket.
3. TIME_WAIT – The socket stays open to handle any packets for the TCP connection that might still be in the network. The TIME_WAIT state can last from 1 to 4 minutes, depending on the implementation.

A WAIT socket can result in significant delays and/or errors for the application or service that relies on that socket. A lot of simultaneous WAIT sockets can prevent applications and services from creating new connections.

Time Wait
In most cases, the FIN_WAIT states are relatively short (in milliseconds) while the TIME_WAIT state is relatively long (1 to 4 minutes). This means that the Wait and Time Wait metrics should be fairly similar. Significant differences indicate a significant network or other delays between the selected node and another host.

### Interface Metrics in the Metric Browser

These metrics measure the number of collisions and errors on the physical interfaces of the node. To view these metrics, drill down in the Metric Browser as follows:

- Application Infrastructure Performance
  - <tier-name>
    - Individual Nodes
      - <node-name>
        - Host
          - Interface

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description / Notes</th>
<th>Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td># Collisions</td>
<td>Number of collision errors seen on all the interfaces, when an interface was blocked from sending a frame due to ethernet collision detection.</td>
<td>KPI</td>
</tr>
<tr>
<td># Receive Drops</td>
<td>Number of packets dropped while receiving by all interfaces on the host</td>
<td>KPI</td>
</tr>
<tr>
<td># Receive Errors</td>
<td>Number of collision errors seen on all the interfaces, when an interface could not receive a frame due to a physical problem on the interface or the connected cable.</td>
<td>KPI</td>
</tr>
<tr>
<td># Transmit Drops</td>
<td>Number of packets dropped while transmitting by all interfaces on the host</td>
<td>KPI</td>
</tr>
<tr>
<td># Transmit Errors</td>
<td>Number of collision errors seen on all the interfaces, when an interface could not send a frame due to a physical problem on the interface or the connected cable.</td>
<td>KPI</td>
</tr>
<tr>
<td># Total Errors</td>
<td>The total number of ethernet and/or physical errors detected on the interface.</td>
<td>KP</td>
</tr>
</tbody>
</table>
## TCP Connection Metrics in Metric Browser

On this page:

- Viewing TCP Flow Metrics
- TCP Flow Metric Descriptions
  - # Client Limited
  - # Client Zero Window
  - # Connection Errors
  - # Connection Requests
  - # Current Established Connections
  - # Data Retransmits
  - # Delayed Acks (Data Piggy Back)
  - # Delayed Acks (Timeouts)
  - # Errors
  - # Fin Errors
  - # Flows (<1KB)
  - # Flows (1k - 10k)
  - # Flows (10k - 100k)
  - # Flows (100k-1MB)
  - # Flows (1MB - 10MB)
  - # Flows (>10MB)
  - # Flows - Handshake (1SD)
  - # Flows - Handshake (2SD)
  - # Flows - Handshake (3SD)
  - # Flows - Lifetime (1SD)
  - # Flows - Lifetime (2SD)
  - # Flows - Lifetime (3SD)
  - # Flows - RTT (1SD)
  - # Flows - RTT (2SD)
  - # Flows - RTT (3SD)
  - # Flows - TCP Data Rxmt
  - # Flows - TCP Resets
  - # Flows - w/o TCP SACK
  - # Flows - w/o TCP Timestamp
  - # IP Fragment Count
  - # Loss Pkts
  - # Nagle Delays
  - # PIE Events
  - # RetransmissionTimeouts
  - # SACK Retransmits
  - # Server Limited
  - # Server Zero Window
  - # Syn Blackholes
  - # Syn Resets
  - # TCP Resets - Established
  - # TCP Resets - Fin
  - # TTL Changes (1 - 2 hops)
  - # TTL Changes (3 - 4 hops)
  - # TTL Changes (>=5 hops)
  - Avg # TTL Hops
  - Delayed Acks Lag (usec)
  - Initial RTT (usec)
  - Latency - RTT (usec)
  - Lifetime (usec)
  - Loss (per mille)
  - Nagle Delays Lag (usec)
  - Pkts per Sec
  - Rx Pkts per Sec
  - Rx Throughput (BPS)
  - TCP Handshake (usec)
Network Visibility agents can calculate an extensive set of metrics based on the TCP flows observed by the network agents. In addition to KPI, PIE, and Troubleshooting metrics, you can view advanced metrics for network elements of interest (tiers, nodes, links, and Connections) in the Metric Browser. Using time-based charts, you can detect and analyze behavior such as:

- TCP flow setup/teardown times and errors
- Distribution of TCP flows for an element by throughput, setup time, lifetime, and round-trip time
- Flows closed by TCP resets
- Distribution of flows based on TCP configuration options (Selective Acknowledgement, Timestamp)

Note the following:

- For these metrics, the Metric Browser uses the terms Connection and Flow interchangeably. See Flows, Links, and Connections.
- Network Agents do not collect metrics for individual Connections by default. The recommended workflow is to enable TCP Flow metric collection only when you need to diagnose a performance issue on an associated node or Connection. See Dynamic Monitoring Mode and Network Visibility.

### Viewing TCP Flow Metrics

To view advanced network metrics, open the Metric Browser and navigate to Application Infrastructure Performance. The Metric Browser shows connection/flow metrics for the following aggregations:

- Application Infrastructure Performance >
  - Advanced Network >
    - Flows >
      - All Connections/Flows for Application
        - tier-name
          - Advanced Network >
            - Flows >
              - Call from <tier> to <service>
                - Call from <node> to <node-or-service>
                  - Individual Connection/Flow (<node> to <service>)
                - All Connections/Flows for Network Link (<tier> to <service>)
              - All Connections/Flows for tier

### TCP Flow Metric Descriptions

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description / Notes</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td># Client Limited</td>
<td>The number of TCP window updates sent by a client node indicating that it is receiving data at a faster rate than the application can process.</td>
<td>KPI</td>
</tr>
</tbody>
</table>
# Client Zero Window
If a client’s TCP buffer is full, it sends Zero as the TCP receive window size to indicate that it cannot receive more data. The data transfer then stops until the client can process the data in its buffer.

A high rate of Client TCP Zero Window messages indicates a problem with either
- The TCP configuration on the node,
- The client-side application using that flow.

# Connection Errors
The sum of Syn Resets + Syn Blackholes + TCP Resets - Established.

# Connection Requests
The total number of connection requests (successful and unsuccessful) sent during the selected time window.

# Current Established Connections
The number of established (set-up) flows/connections.

# Data Retransmits
Number of TCP data packets that were retransmitted for all TCP Connections.

# Delayed Acks (Data Piggy Back)
The average number of times a receiving node sent an ACK (acknowledgment) by "piggybacking" the ACK onto another message.

# Delayed Acks (Timeouts)
The average number of times a receiving node sent an ACK (acknowledgment) because it's Delayed ACK timer expired.

This is a "worst-case" delay for the Delayed ACK algorithm and occurs most often when Nagle's algorithm and Delayed ACK are enabled on the sending and receiving node (respectively). A high rate of delayed ACKs can contribute significantly to the average Latency on a Connection.

# Errors
The number of TCP messages sent indicating an error in setting up the connection (SYN errors) or tearing down the connection (FIN errors).

# Fin Errors
The number of errors seen while tearing down the connections (TCP FIN errors). A lot of connections in FIN wait states can cause delays in creating new connections.

# Flows (<1KB)
Use these metrics to analyze the distribution of flows by throughput.

# Flows (1k - 10k)

# Flows (10k - 100k)

# Flows (100k-1MB)

# Flows (1MB - 10MB)

# Flows (>10MB)
| # Flows - Handshake (1SD)                        | The number of TCP flows with connection-setup times that are 1, 2, or 3 Standard Deviations outside (higher or lower than) the average for that connection group. Under ideal conditions, all flows should be within 1SD of the average.                                                                                   | Advanced |
| # Flows - Handshake (2SD)                                      |                                                                                                                                       |          |
| # Flows - Handshake (3SD)                                      |                                                                                                                                       |          |

| # Flows - Lifetime (1SD)                                      | The number of TCP flows with lifetimes that are 1, 2, or 3 Standard Deviations outside (higher or lower than) the average. Under ideal conditions, all flows should be within 1SD of the average. 2SD or 3SD flows indicate inconsistent flow treatment along the network path. These inconsistencies can occur when a network service decreases the available bandwidth intentionally *(bandwidth throttling)* or prioritizes some types of traffic over others *(traffic shaping)*. Short-lived connections, even if they are intermittent, can indicate an issue worth investigating. The more short-lived connections get generated, the more resources are spent setting up and tearing down these connections. | Advanced |
| # Flows - Lifetime (2SD)                                      |                                                                                                                                       |          |
| # Flows - Lifetime (3SD)                                      |                                                                                                                                       |          |

| # Flows - RTT (1SD)                                           | Number of flows whose Round Trip Times are 1, 2, or 3 Standard Deviations higher than the average for all flows in the parent group. Under ideal conditions, all flows should be within 1SD of the average. High-RTT connections, even if they are intermittent, can indicate an issue worth investigating. If the average response time for an ecommerce web app is 2 seconds (acceptable), but 5% of transactions are 20 seconds or higher (not acceptable), this can result in a significant number of unhappy customers and lost revenue. | Advanced |
| # Flows - RTT (2SD)                                           |                                                                                                                                       |          |
| # Flows - RTT (3SD)                                           |                                                                                                                                       |          |

| # Flows - TCP Data Rxmt                                      | Number of flows within which any packet was retransmitted. Retransmissions are an indication of packet loss.                                                                                       | Advanced |
| # Flows - TCP Resets                                         | The average number of flows that were closed by a TCP Reset.                                                                                                                                     | KPI      |
| # Flows - w/o TCP SACK                                      | Number of flows with the TCP Selective Acknowledgment (SACK) option disabled. With SACK enabled, a receiver can send SACK packets to acknowledge receipt of multiple data packets in the case of lost segments. This improves network performance by reducing the number of retransmissions. With SACK disabled, the receiver must resend all the packets after the last lost segment even if they were received by the peer. | Advanced |
| # Flows - w/o TCP Timestamp                                  | Number of flows with the TCP Timestamp option disabled. This option is used for calculating more accurate Round Trip Times.                                                                          | Advanced |
| # IP Fragment Count                                          | The total number of IP fragments attributes to this TCP connection group. A high level of fragmentation is an indication of network issues and can severely affect application performance.                                                                    | KPI      |
| # Loss Pkts                                                  | The total number of packets that were lost (sent but never received).                                                                                                                             | Advanced |
| # Nagle Delays                                               | The number of times a message-send event was delayed because the sending node had to wait for previously-sent data to be acknowledged (ACK'd). This occurs most often when Nagle's algorithm and Delayed ACK are enabled on the sending and receiving node (respectively). | Advanced |
PIE Events | Performance Impacting Elements (PIE) are useful for identifying the location of actual or potential bottlenecks:
---|---
Client Limited and Client Zero Window events indicate a possible problem on the client node.
RTOs indicate a possible problem on the network path between two nodes.
Server Limited and Server Zero Window events indicate a possible problem on the server node.

RetransmissionTimeouts | Retransmission Timeouts (RTOs) are a sign of network packet loss, which results in retransmission of data when the TCP retransmission timer expires (Timer to make sure data is ACK'ed).
---|---
- Typically this timer varies from 200ms-3 sec by default (different for each operating systems and their versions).
- It causes severe performance degradations as a considerable amount of time gets wasted to resend the lost data.
- TCP falls back to the "Slow Start" phase impacting the performance even more.

SACK Retransmits | Average number of packets that were retransmitted due to a SACK message that indicated an unreceived packet.

Server Limited | The number of TCP window updates sent by a server node indicating that it is receiving data at a faster rate than the application can process.

Server Zero Window | If a server’s TCP buffer is full, it sends Zero as the TCP receive window size to indicate that it cannot receive more data. The data transfer then stops until the server can process the data in its buffer.

- A high rate of Server TCP Zero Window messages indicates a problem with either
  - The TCP configuration on the node,
  - The client-side application using that flow.

Syn Blackholes | The number of connection attempts that went unanswered and resulted in a failure. Syn blackholes can severely impact application performance.

Syn Resets | The number of connection attempts that were explicitly refused by the other host. Syn resets can severely impact application performance.

TCP Resets - Established | The average number of times an established TCP flow was reset.

TCP Resets - Fin | The average number of times a TCP flow was reset while it was in the process of getting closed.

TTL Changes (1 - 2 hops) | The number of routing-hop changes experienced by this connection. Frequent variations in routing-hop changes indicate routing problems in the network and can severely affect app performance.

- Under ideal conditions, the number of hops should be consistent.

Avg # TTL Hops | The average number of routing-hop changes experienced by this connection.

Delayed Acks Lag (usec) | The average amount of lag that Delayed ACKs are adding to the overall Latency of the Connection.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial RTT (usec)</td>
<td>Round trip time for the initial two SYN packets (between SYN and SYN-ACK or SYN-ACK and ACK depending upon whether the agent is running on client or server).</td>
<td>KPI</td>
</tr>
<tr>
<td>Latency - RTT (usec)</td>
<td>Average round-trip latency (from packet transmission to acknowledgment) for all packets.</td>
<td>KPI</td>
</tr>
<tr>
<td>Lifetime (usec)</td>
<td>Average lifetime (from initial setup to final teardown) for TCP sessions in the Connection group.</td>
<td>KPI</td>
</tr>
<tr>
<td>Loss (per mille)</td>
<td>The number of packets lost per 1000 packets sent. “Per mille” is a percentage with one additional digit of precision.</td>
<td>KPI</td>
</tr>
<tr>
<td>Nagle Delays Lag (usec)</td>
<td>The average amount of lag that Nagle delays are adding to the overall Latency of the Connection.</td>
<td>Advanced</td>
</tr>
<tr>
<td>Pkts per Sec</td>
<td>Average rate of packets sent and received</td>
<td>KPI</td>
</tr>
<tr>
<td>Rx Pkts per Sec</td>
<td>Average rate of packets received</td>
<td>Advanced</td>
</tr>
<tr>
<td>Rx Throughput (BPS)</td>
<td>Average rate of bytes received</td>
<td>Advanced</td>
</tr>
<tr>
<td>TCP Handshake (usec)</td>
<td>Round trip time for the initial three-way connection setup, for all flows in the parent group:</td>
<td>KPI</td>
</tr>
<tr>
<td></td>
<td>1. SYN (client --&gt; server)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. SYN-ACK (client &lt;-- server)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. ACK (client --&gt; server)</td>
<td></td>
</tr>
<tr>
<td>Throughput (BPS)</td>
<td>Throughput (bytes per second) for the application of interest on all TCP sessions.</td>
<td>KPI</td>
</tr>
<tr>
<td>Tx Pkts per Sec</td>
<td>Rate of packets sent</td>
<td>Advanced</td>
</tr>
<tr>
<td>Tx Throughput (BPS)</td>
<td>Rate of traffic received</td>
<td>Advanced</td>
</tr>
</tbody>
</table>
FAQs for Network Visibility

On this page:

- General
- Logs

General

Why are there no connection metrics for a tier, node, or network link?

Click here to expand...

Network Agents do not collect connection metrics by default. The recommended workflow is to identify the link with the network issue and configure the relevant agents to collect metrics for the relevant connections. See Dynamic Monitoring Mode and Network Visibility.

The Network Agent cannot register with the Controller. What should I do?

Click here to expand...

If a Network Agent cannot register with the Controller, do the following:

- Check that the user account has a Network Visibility product license.
- If the user account has license rules defined, make sure these have the correct number of license units allocated. To change the number of allocated units in a rule:
  - Go to Controller Settings (gear icon) > License > Rules.
  - Edit the License Rule of interest. (There might be only one License Rule, named Default.)
  - In the General tab, set the Allocated Units field for the Network Visibility license and apply the change.

In some cases, I see that an application flow for a JMS queue goes in one direction but a TCP connection used by that queue goes in the opposite direction. Why is this?

Click here to expand...

In most cases, the network links and TCP connections used by an application flow have the same direction (source --> destination) as the flow itself. You might see different directions, however, if two tiers transfer data via a JMS queue. In some JMS implementations, the individual nodes in each tier initiate the TCP connection to the queue, so the direction is always:

node (source) --> queue (destination)

Some of these connections might be used by an application flow in the opposite direction:
queue (source) --> tier (destination)
How do I change the Network Agent communications port?

Click here to expand...

When you start the agent, the appd_netmon process spawns the appd-agent process. These two processes communicate over TCP port 3892 by default. If this port is already in use, you should see a log message about this in one or both files. To configure the agent to use a different port, do the following:

1. Use the `netstat` command to verify that the new port is not in use.
2. Update the Network Agent:
   a. Open the following file in a text editor: `<network_agent_home>/conf/agent_config.lua`
   b. Set the `port` option to the new TCP port number (under `webserver_config`).
      ```lua
      webserver_config = {
         port = <new-port-number>,
         request_timeout = 10000,
         threads = 2,
      }
      ```
   c. Save the file and restart the Network Agent.
3. Update the App Agent:
   a. Open the following file in a text editor: `<app_agent_home>/<version-number>/external-services/netviz/netviz-service.properties`
   b. Set the `netviz.agent.api.service.port` option to the new TCP port number.
   c. Save the file and restart the App Agent.

The Application I want to Monitor uses TCP Port 32768 or Higher. How do I configure the Network Agent to Monitor this Port?

Click here to expand...

1. Open the following file in a text editor: `<network_agent_home>/conf/agent_config.lua`
2. If you plan to monitor any application or service that uses any TCP ports higher than 32767, uncomment the `application_service_ports` block and specify these ports as a comma-separated list in the `ports` option:

   Original:
   ```lua
   --{}[
   application_service_ports = {
   ```
2.

```lua
ports = "",
}
--]]
Edited:

application_service_ports = {
ports = "40000, 41000",
}
```

**Logs**

**How do I resolve the error, "ERROR MsgZmq::Bind: zmq_bind failed: File name too long"?**

- Click here to expand...
  
  This error occurs when the file name is too long. The complete path to the file is considered as the file name. To resolve this error, move the Network Agent to a smaller absolute path.

**Why do I see the error, "ERROR cw_flowgroup_uri_cb: Failed to write data on connection"?**

- Click here to expand...
  
  This error occurs when Network Agent supports a large number of AppServer Agents. If you notice the error frequently, restart the agent with increased number of thread counts in `webserver_config` in `agent_config.lua` config file.

**Why do I see the error, "ERROR ip_flowgrp_lookup: flowgrp alloc failed"?**

- Click here to expand...
  
  Connections (Source IP : Source Port : Destination IP : Destination Port) in Network Agent are represented as flows. Aggregating these connections on a source port gives flow group, represented as (Source IP : Destination IP : Destination Port). The Network Agent can monitor only a certain number of flow groups. This error occurs when maximum number of flow groups is reached and the Network Agent cannot monitor any new incoming flow groups.

**Why do I see the error, "DEBUG adns_resolve: ip resolve error: Name or service not known"?**

- Click here to expand...
  
  AppDynamics resolves IPs to provide fully qualified domain name (FQDN) information to the controller for the IPs that are seen in the system. This error occurs when some IP addresses are not resolved.
Network Visibility Concepts

The following concepts are central to Network Visibility:

- Network Visibility can identify two types of performance bottlenecks: network-performance bottlenecks and app/network-interaction bottlenecks (i.e., issues in the way that an application tier or node uses network resources).
- The Network Dashboard shows the network-level view of a monitored application. You can quickly identify network elements with performance issues (red/yellow links or tiers), view Key Performance Indicators for each network element, and troubleshoot individual elements (right-click on an element and choose View Metrics).
- Network agents can detect load balancers and TCP endpoints, which appear in network flow maps (but not application flow maps) and sometimes act as traffic sources and destinations.
- Network Visibility can collect metrics for individual application flows, TCP connections, and network links.
- To support in-depth troubleshooting, you can configure network agents to collect detailed diagnostic metrics for nodes and connections with performance issues.
Identifying Performance Bottlenecks

Performance Bottleneck Categories

Network Visibility can pinpoint performance bottlenecks down to a specific tier, node, or network path. Either of the following can slow down an application significantly:

- **Application/Network bottlenecks**
  Many bottlenecks identified as "network issues" actually have their root cause in the way that an application or a server utilizes network resources. Network Visibility can detect many different types of app/network bottlenecks as well as bottlenecks due to TCP socket issues on a specific node.

- **Network bottlenecks**
  A network bottleneck indicates that a specific section of the network cannot handle the application loads between the source and destination nodes. Symptoms of a network bottleneck include a high rate of packet loss or high round-trip times (RTTs) between two nodes.
Load Balancers and TCP Endpoints

You might see objects in a Network Flow Map that do not show up in an Application Flow Map. This is because these Network Flow Map objects represent devices that are detected by the network agents, and are relevant to the monitored application, but the detected devices do not have network agents installed.

Load Balancers

Network Visibility detects load balancers automatically and displays them in Network Flow Maps. Load balancers seek to optimize the distribution of network traffic and often hide internal IPs from external nodes. Application Flow Maps do not show load balancers because these devices do not act as application nodes.

An intermediate load balancer (LB) has the effect of splitting an application message into two separate connections, the nodeA connection to a Load Balancer and the Load Balancer connection to node B,  (nodeA -> LB and LB -> nodeB). This additional layer of detail in the Network Browser makes it easier to identify network segments that have specific performance issues.

TCP Endpoints

The Controller uses TCP endpoints to represent flows where the monitoring vantage point is from one end. The network agent monitors the connection between a monitored node and a TCP endpoint. However, the network agent cannot determine if this endpoint is on another node or on an intermediate device, such as a load balancer, that splits the connection in two.

The screenshots show two different views of the same application:

The Application Dashboard shows two tiers connecting to two MySQL instances. The Application Flow Map shows application messages between monitored nodes that are directly visible to app agents.
The **Network Dashboard** shows TCP endpoints on links between tiers/nodes. The Network Flow Map shows network agents installed (left) and nodes that the network agents cannot see directly (right).

In the following example, the network agent monitors a connection between an Inventory-Tier node (left) and an Inventory Database service. A TCP endpoint divides the link into a colored line representing the connection visible to the network agent, and a gray line representing a connection that is not instrumented by Network Visibility. A gray line connection link without Network Visibility means one of two things:

- The TCP endpoint and an Inventory Database service are *on the same device* with an intermediate device that splits the connection in two.
- The TCP endpoint and an Inventory Database service are *on two devices* using a separate connection that doesn't allow for network visibility.

For these reasons, gray line connection links do not have associated metrics.
Flows, Links, and Connections

On this page:
- Application Flows
- Network Links
- TCP Connections

Network Visibility can collect metrics for the following object types:
- Network links between tiers, or between tiers and load balancers/TCP endpoints
- Connections between nodes, or between nodes and load balancers/TCP endpoints

Troubleshooting a performance issue in Network Visibility is often a matter of "drilling down" from an application issue (flow) to a network issue (link) to the root cause (connection). See Workflows and Example Use Cases.

Application Flows

App Agents collect metrics for application flows between tiers, as shown in the Application Flow Map. This diagram shows the calls, transactions, and messages for Application X between Tiers 1 and 2.

Network Links

Network Agents collect metrics for network links, which you can view in the Network Dashboard. A network link is the set of all traffic between two tiers, or between a tier and an intermediate load balancer or TCP endpoint. If an application flow transports traffic via a load balancer, Network Visibility creates a link on each side and calculates metrics for each link. This diagram shows the network traffic for Application X sent and received by the two tiers via a load balancer.
TCP Connections

Individual nodes send and receive traffic over connections. In Network Visibility, a Connection is a set of all traffic with the same Source IP, Destination IP, Destination Port, and Protocol. This diagram shows the individual Connections that the four nodes use to send and receive traffic for Application X via the load balancer.
Dynamic Monitoring Mode and Network Visibility

On this Page:
- Changing the DMM on a Network Agent
- Connection Diagnostics and Network Bottlenecks
- Example Workflow

Related pages:
- Dynamic Monitoring Mode and Server Visibility
- Network Visibility Metrics

Like Server Visibility agents, network agents support Dynamic Monitoring Mode (DMM). Rather than have all network agents report all metrics all the time, you can run each agent in one of three modes:

- KPI Mode – Network KPI metrics for all monitored objects (application flows, tiers, nodes, and network links) except individual connections
- Diagnostic Mode – Network KPI metrics for all monitored objects, including connections
- Advanced Diagnostic Mode – All network metrics for all monitored objects, including connections

Every Network Visibility metric has a default DMM class (KPI, Diagnostic, and Advanced Diagnostic). To see the DMM class for each metric, see Network Visibility Metrics.

All network agents run in KPI mode by default. The recommended workflow is to

1. Run all network agents in KPI mode.
2. When you see a performance issue on a specific node or network link, increase the metric level on the associated network agents to Diagnostic. Collect KPI metrics for the connections.
3. Identify the connections with performance issues, based on the connection KPIs.
4. To troubleshoot an individual connection, increase the metric level on the associated network agents to Advanced Diagnostic. Collect advanced metrics for the connection.
5. When the issue is resolved, reset the agents back to KPI mode.

Changing the DMM on a Network Agent

1. Click the gear icon (⚙️) in the top-right corner of the Controller page, choose AppDynamics Agents, and go to the Network Visibility Agents table.
2. Select the agents of interest, right-click, and choose Change Dynamic Monitoring Mode.

The network agent retains its DMM setting even if the agent is stopped and restarted. See Managing Network Agents in the Controller.

Connection Diagnostics and Network Bottlenecks

Network agents can collect an extensive set of individual Connection metrics within a network link. Some TCP performance issues that Network Visibility can detect include:

- The data-receive window is too small (or zero) on the client or the server, which slows down the transfer of data.
- One or more servers are experiencing errors setting up or tearing down the connection for an individual TCP session.
- The client and server take a long time to set up an individual session.
- The application is using a lot of short-lived connections. TCP is most efficient when long, stable connections are used.
- Some TCP sessions have unusually high round-trip times (RTTs). When TCP is performing well, RTTs are stable and determined mostly by the network path between two nodes.

The network agent does not collect any Connection metrics in KPI mode (the default setting). You can change the Dynamic Monitoring Mode on a network agent when you need to diagnose a node or network path that is monitored by that agent.

Example Workflow
After you complete the initial setup, you can set the Dynamic Monitoring Mode on individual network agents as needed. An example workflow might look like the following:

- The DevOps team for a large enterprise monitors its IT infrastructure using network agents on critical servers. All agents are initially set to KPI mode.
- The Network Dashboard shows a spike in latency on a network link between tier-A and tier-B.
- A DevOps team member does the following:
  - Sets the network agent DMM on the tier-A and tier-B servers to Diagnostic.
  - Collects KPI metrics for the Connections between the tier-A and tier-B nodes.
  - Identifies one high-latency connection between nodes TA-N1 and TB-N3, based on the Connection KPIs. The KPIs for all other connections are within acceptable bounds.
  - Sets the network agent DMM on TA-N1 and TB-N3 to Advanced Diagnostic and the DMM on all other agents to KPI.
  - Notes a set of spikes in Nagle delays on this Connection. These spikes correspond to the latency spikes she noticed on the network link.
  - Reconfigures TCP on the two nodes and monitors the Connection. The Nagle's and latency spikes no longer occur.
  - Resets DMM on the TA-N1 and TB-N3 agents back to KPI.
Workflows and Example Use Cases

On this page:
- Application Troubleshooting
- Transaction Diagnosis
- Network Health Check

Network Visibility makes it easy to answer the question: *Is the network impacting my application’s performance?* This section describes the high-level troubleshooting workflows and includes several examples of how to use these workflows to solve real-world network problems.

**Application Troubleshooting**

*I’m in the Application Dashboard and I see slow, very slow, error, or stalled transactions. Are network conditions to blame?*

- Application Dashboard: Are network conditions to blame for the issue?
- Network Dashboard: Which network tiers and links are associated with these network conditions?
- Basic Diagnosis
- Network Dashboard (Advanced Diagnosis): On which TCP connections are these network issues occurring?

**Application Dashboard: Are network conditions to blame for the issue?**

Network agents collect network-performance metrics for each application tier. If you see performance issues in the Application Flow Map, open a popup and click the Network tab. Do you see correlations between the Response Time and the other charts? If so, it indicates that the Response Time is affected by

- TCP effects between the two tiers (Latency).
- Performance Impacting Events at the client tier, server tier, or the network path between the tiers (PIE).
- Errors setting up or tearing down TCP connections (Errors).

**Network Dashboard: Which network tiers and links are associated with these network conditions?**

Go to the Network Dashboard, which shows the TCP and network context for the application. Each tier and link has a View Metrics right-click option that opens a context-sensitive dashboard. The dashboard charts make it easy to find network root causes. Start from the top-left chart in each dashboard, then look for metric correlations in the other charts.
The tier dashboard answers the question: *Which network metrics on this tier correlate with my outlier transactions?*

The link and tier dashboards both have a Network Pie - Contributors chart, which answers the question: *Where are these network conditions occurring – on the client tier, server tier, or the network link?*

In this example, the top chart shows that a spike in Stalled transactions is correlated with a spike in PIE (Performance Impacting Events); the bottom chart shows that these events occurred on the client tier. Given this information, you can then find other correlated metrics and identify the issues on the client tier.
Basic Diagnosis

You now have a set of correlated network metrics and associated tiers/links. In many cases, this information is enough to identify and resolve the issue. You can contact your network team and say: I see the following network issues on tier X, link Y, etc. and these issues correlate with performance issues in my application. You can send them a URL to the Network Dashboard and say: Click on link Y and look at the connections. The issues are happening on one or more of these connections. This is often enough information for the networking team to identify and resolve the issue.

Network Dashboard (Advanced Diagnosis): On which TCP connections are these network issues occurring?

If you need to collect more information, you can drill down to the TCP connection(s) where the network conditions are occurring. Click on the relevant network link and choose Connections in the popup. Sort the table based on the KPI metrics for the monitored connections (Throughput, TCP Loss, PIE, Errors). You can then right-click on a connection and view metrics in the Metric Browser or in a context-sensitive dashboard.

Network Agents do not monitor individual Connections by default. The recommended workflow is to enable Connection monitoring only after you narrow an issue down to a specific set of links. See Dynamic Monitoring Mode and Network Visibility.
Transaction Diagnosis

I'm troubleshooting a specific outlier (Slow, Very Slow, Error, or Stalled) transaction. Are network conditions to blame?

- Transaction Snapshots list: Is the network to blame for a specific slow, stalled, or Error transaction?
- Transaction Snapshot viewer: Where (at which tier) are the delays, stalls, or errors occurring?
- Network Dashboard for Transaction Snapshot: Are there any correlations between problematic transactions and network-performance issues around the time of this transaction?

Transaction Snapshots list: Is the network to blame for a specific slow, stalled, or Error transaction?

You can diagnose the network performance for an individual transaction in the transaction viewer. Double-click on a snapshot in a Transaction Snapshots list.

Transaction Snapshot viewer: Where (at which tier) are the delays, stalls, or errors occurring?

When the Transaction viewer appears, click Drill Down on the tier where the delays, stalls, or errors are occurring.

In the Tier Overview tab, click Network.

Network Dashboard for Transaction Snapshot: Are there any correlations between problematic transactions and network-performance issues around the time of this transaction?

The transaction dashboard shows the rate of transaction outliers, Performance Impacting Events, and Network Errors when the transaction occurred. Each chart has the Snapshot Timerange highlighted. The Network Impact on Transactions chart (top left) highlights any correlations between transaction outliers, Network Errors, and Performance Impacting Events. You can then look for correlations in other charts and "drill down" to the root cause.
Network Health Check

I don't see any application issues right now, but I'd like to monitor my network and identify any potential issues before they start to affect my applications.

- Network Dashboard: Is any tier or link performing outside its baseline?
- Network Dashboard: Do any network links or connections have significant performance issues?

**Network Dashboard: Is any tier or link performing outside its baseline?**

To perform a quick network health check, go to the Network Dashboard and enable baselining. Elements with network KPIs that exceed the baseline for that element are colored yellow or red.
If the network KPIs for a tier or link are outside the performance baseline for that element, you can view baselines and outliers for that element in the Network tab of the popup. If an element has KPI outliers, you can also diagnose that element in a context-sensitive dashboard (right-click and choose View Metrics).

**Network Dashboard:** Do any network links or connections have significant performance issues?

To see immediately if any links or connections have performance issues, open the Connections Explorer (click the Explorer link to the immediate right of the Network Flow Map). Here you can see the KPIs of all network links and connections in one table. You can sort the table based on KPIs or Errors and expand each network-link to view KPIs for all connections within that link.
Example Use Cases

- Network Congestion - Example Use Case
- Packet Loss - Example Use Case
- TCP Port Exhaustion - Example Use Case
- TCP Window Bottlenecks - Example Use Case
- Traffic Throttling - Example Use Case
Network Congestion - Example Use Case

On this page:
- Application Symptoms
- Network Diagnosis

Network congestion can have a number of root causes, such as
- A user might launch a "bandwidth-hogging" backup job during a period of peak network utilization.
- A server might drop packets at seemingly random intervals due to a TCP misconfiguration or other issue.

Traditional monitoring tools such as SNMP are good at detecting persistent network bottlenecks but often miss intermittent bottlenecks. With Network Visibility, you can easily identify and diagnose intermittent network bottlenecks that affect application performance.

Application Symptoms

A DevOps engineer is responsible for monitoring a mission-critical app. One day, she opens the Application Dashboard and sees that Ecom-Tier1 has suddenly gone partially red, which indicates performance degradation on one of the nodes in this tier. She also sees a small spike in response times for the overall application. She decides to investigate.

Network Diagnosis
1. To determine if there is a network problem, she does the following:
   a. Goes to the Transaction Snapshots page and filters the list to show only stalled calls.
   b. Double-clicks on a specific call to view the transaction flow map. Because it was a stalled call, the entire flow map doesn’t appear.
   c. Clicks Drill Down to open the Transaction Dashboard and then switches to the Network tab to analyze the network performance during this call.

2. The Network Impact on Transactions chart (top left) shows a spike in Performance Impacting Events (PIE) around the time of the transaction snapshot. By showing only PIE and Stalls, she can see that the stalled calls and spike in PIE occur within the same time window.

3. In the Network Pie - Contributors chart, she can see that all the Performance Impacting Events are retransmission timeouts (RTOs). This indicates packet loss on the network paths between the two tiers, which is confirmed by the spike in TCP Loss.
that occurs within the same time window.

4. She knows there is a correlation between stalled calls and retransmissions. Now she wants to know more about these events. The Retransmissions Per Min chart indicates that all these are all data retransmits, which indicate a problem on the network path between two nodes. (SACK retransmits indicate a problem setting up TCP connections on one or two nodes.)

5. In a few seconds, she has identified a chain of correlation: stalled calls > PIE > Retransmissions > Data retransmissions on a network path. The next step is to identify where in the network these retransmissions are occurring. She switches over to the Network Dashboard and looks for network links with high PIE scores. She sees the PIE scores on two links are 59 – much higher than any other links. Both of these links are used by ECom-Tier1 and Ecom-Tier2 to connect with Order-Tier via a load balancer.
5. Given this information, she decides to investigate one of these tiers. She right-clicks on Ecom-Tier1 and chooses View Metrics. The Dynamic Dashboard for this tier appears. The dashboard shows the same metric correlations for the entire tier that she saw for the individual transaction.
7. Now that she’s confirmed that data retransmissions are spiking for the entire tier, she wants to identify the TCP connections on which these events are occurring. She clicks on the network link between Ecom-Tier and the load balancer. She can see that that PIE is happening on both connections:

ECOM_T1N1:8080 < – > Load_Balancer_10.0.11.77:8080
ECOM_T1N2:8080 < – > Load_Balancer_10.0.11.77:8080

8. Given this information, she contacts the network-management team in her organization and says: “I can see that there was intermittent loss from 9:07-9:10 AM on the following TCP connections, and that these correlate with a spike in stalled calls.” The network team can now investigate the network paths used by these connections and see if the problem is on the load balancer or elsewhere.
Packet Loss - Example Use Case

In some cases, packet loss can occur on a server or intermediate device. The problem might be hardware-related — perhaps the device simply cannot handle peak traffic loads, or there might be faulty cabling on one or more network interfaces. The problem might be software-related — a bug or misconfiguration that results in intermittent packet loss at seemingly random intervals. When packet loss affects application performance, you can use Network Visibility to pinpoint the nodes, devices, and TCP connections where the packets are getting dropped.

Application Symptoms

A DevOps engineer is responsible for monitoring the performance of a mission-critical app. One day, she opens the Application Dashboard notices that the Order-Tier, and the application flows to and from this tier, have suddenly turned red. She decides to investigate.

Network Diagnosis

1. She switches over to the Network Dashboard and sees that the Order-Tier and Payment-Tier, and the network link between them, show a spike in PIE and network errors.
1. She decides to analyze an individual transaction. She goes to the Transaction Snapshots page and double-clicks on a Very Slow transaction. The Transaction Overview shows that 97% of the delay is on the link between Order-Tier and Payment-Tier. She drills down into the Order-Tier and goes to the Network tab.

2. She decides to analyze an individual transaction. She goes to the Transaction Snapshots page and double-clicks on a Very Slow transaction. The Transaction Overview shows that 97% of the delay is on the link between Order-Tier and Payment-Tier. She drills down into the Order-Tier and goes to the Network tab.
3. Scanning the dashboard for this transaction, she sees immediately that
   a. The Network Impact on Transactions graph (top left) shows a spike in PIE and Very Slow Calls around the snapshot time range.

   ![Network Impact on Transactions](image)

   Network Impact on Transactions

   ![Network Impact on Transactions](image)

   - PIE
   - Slow Calls
   - Very Slow Calls
   - Stalls
   - Errors
   - Network Errors

   b. She sees the correlation between Errors and Very Slow Calls, so she looks at the Network Errors - Contributors chart. She sees immediately that all these errors are Syn Blackholes. The Order-Tier node is trying to establish connections to the Payment-tier node, but something in the middle is silently dropping the connection. When this happens, it's usually a firewall or other intermediate device that is
dropping packets. The requesting tier tries and retries to re-establish the connection, which introduces significant delays.

![Network Errors - Contributors](image)

3. The Retransmissions Per Min chart confirms that the retransmissions during the time window of interest are all Data Retransmits, which means that the drops are occurring on an intermediate device rather than at an application node.

![Retransmissions Per Min](image)

4. Given this information, she now wants to pinpoint the specific TCP connections where the packet drops are occurring. She enables Advanced Diagnostic monitoring on the relevant network agents to collect TCP diagnostic metrics. She then goes to the Network Dashboard and clicks on the links between Order-Tier and Payment-Tier. This provides her with all of the connections with elevated PIE and errors.

<table>
<thead>
<tr>
<th>Source Tier / Node</th>
<th>Destination Tier / Node</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Destination Port</th>
<th>Throughput (Kbps)</th>
<th>PIE</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORD-N1_1</td>
<td>PAY-N1_0</td>
<td>10.0.11.100</td>
<td>10.0.31.100</td>
<td>8080</td>
<td>96.0</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>ORD-N1_0</td>
<td>PAY-N1_0</td>
<td>10.0.11.100</td>
<td>10.0.31.100</td>
<td>8080</td>
<td>96.0</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>ORD-N5_1</td>
<td>PAY-N1_0</td>
<td>10.0.11.101</td>
<td>10.0.31.100</td>
<td>8080</td>
<td>48.6</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>ORD-N5_0</td>
<td>PAY-N1_0</td>
<td>10.0.11.101</td>
<td>10.0.31.100</td>
<td>8080</td>
<td>48.6</td>
<td>1</td>
<td>18</td>
</tr>
</tbody>
</table>
Given this information, she contacts the network-management team in her organization and says: "We're seeing intermittent SYN blackholes and retransmissions on these connections, can you please investigate and fix."
TCP Port Exhaustion - Example Use Case

On this page:
- Application Symptoms
- Network Diagnosis

A TCP socket is a specific TCP port on a specific node. Port exhaustion occurs when a node runs out of available ports. When an application stops using a specific port, the port enters a "time-wait state" before it becomes available for use by another application. If a node runs out of available ports to create new connections, the symptom appears as a spike in application errors.

Application Symptoms

A DevOps engineer is responsible for monitoring the performance of a mission-critical app. She opens the Application Dashboard one day and sees a sudden spike in transaction errors on the Ecom-Tiers and for the entire application. She also sees that response times for the entire app are going down and load is increasing. She decides to investigate.

Network Diagnosis

1. She switches over to the Network Dashboard, right-clicks on Ecom-Tier2, and chooses View Metrics. The Host Stack KPIs show an increase in Wait Sockets at the same time as the increase in Error transactions.
1. She knows there are a lot of Wait sockets for the entire tier. Now she wants to pinpoint the specific node (or nodes) where these wait sockets are occurring. She returns to the Network Dashboard, clicks on Ecom-Tier2, and clicks on ECOM_T2N1 in the Nodes tab.

2. She knows there are a lot of Wait sockets for the entire tier. Now she wants to pinpoint the specific node (or nodes) where these wait sockets are occurring. She returns to the Network Dashboard, clicks on Ecom-Tier2, and clicks on ECOM_T2N1 in the Nodes tab.
3. The TCP Socket chart (right) for node ECOM_T2N2 shows that the level of wait sockets is in an acceptable range. She switches over to view ECOM_T2N1.

4. The TCP Socket chart for this node shows a significantly high number of Wait sockets. She knows now that the spike in application errors is occurring because this node is running out of available sockets.
TCP Window Bottlenecks - Example Use Case

1. Receiver has a TCP Receive buffer for incoming packets. It advertises the available buffer space to the sender.
2. Sender sends a chunk of data based on the advertised window size.
3. Application on the receiver processes the packets in the buffer and frees up space in the buffer.
4. Receiver advertises the new window size.

This process works smoothly unless the receiving node cannot process incoming packets quickly enough. In this case, the receiver sends TCP Limited or TCP Zero messages to the sender and the transfer slows down.

Application Symptoms

A DevOps engineer is responsible for monitoring a mission-critical app. She scans the Application Dashboard and notices that a link and tier have turned yellow and another link has turned red. She decides to investigate.
Network Diagnosis

1. She switches over to the Network Dashboard and sees a significant increase in Performance Impacting Events (PIE) between the Order-Tier and the Payment-Tier. While the Application Dashboard shows performance issues on the upstream Order-Tiers, the Network Dashboard implies that the problem is further downstream—between the Order-Tier and
2. To troubleshoot further, she goes to the Transaction Snapshots list, filters on stalled transactions, and double-clicks on a transaction to drill down.

3. She drills down into the Order-Tier, since nearly all of the transaction time (99.7%) occurred at this tier.
4. In the Transaction Drilldown page, she switches over to the Network view. Scanning the dashboard, she can see immediately that
   a. The transaction correlates with a spike in stalled calls and Performance Impacting Events.
   b. All of these events (Client Limited and Client Zero Window) took place on the client (Order-Tier) node.
5. She returns to the Network Dashboard, right-clicks on the Order-Tier, and chooses View Metrics. She immediately sees that the Client Limited and Client Zero Window events are spiking for the entire tier and correlate with the spike in stalled calls.
Traffic Throttling - Example Use Case

Firewall throttling occurs when a firewall or other intermediate device prioritizes some connections over others, or denies some connections altogether. This might be due to traffic policies explicitly defined on the device or to one or more misconfigurations.

Application Symptoms

A DevOps engineer is responsible for monitoring the performance of a mission-critical app. One day she scans the Application Dashboard and notices that

- Ecom-Tier1 and Ecom-Tier2 are showing a lot of errors.
- Traffic Loads and Errors are going up while response times are going down (bottom charts).

Network Diagnosis

1. She switches over to the Network Dashboard and sees immediately that a lot of network errors are occurring on the links between the Ecom-Tiers and the load balancer in the center.
2. She right-clicks on Ecom-Tier1, chooses View Metrics, and sees that
   a. The Network Impact on Transactions chart shows that transaction Errors and Network Errors have started going up at the same time.

   ![Network Impact on Transactions](image)

   b. The Network Errors - Contributors chart shows that two types of Network Errors are increasing:
      i. Syn Resets — This reset occurs when the firewall explicitly rejects a connection request before it can get established.
ii. **RST on Established** — This reset occurs when the firewall shuts down an established connection due to traffic-throttling or other policy on the device.


iii. The Connection Rate Info chart shows that the rate of Connection Errors and Resets are exactly the same — in other words, every connection error is a connection reset. This shows that the firewall in the load balancer is actually rejecting connections.
User Interface

This section describes the following:

- Network Dashboard
- Connection Explorer
- Custom Dashboards for Network Visibility
The Network Dashboard provides a network-layer view of your application. Here you can quickly see if any part of the network is impacting application performance. The bottom charts show the overall network health. Each tier and link shows network KPI (Key Performance Indicator) metrics. Network Agents detect intermediate load balancers and TCP endpoints. You can do the following:

- Enable baselining to visualize network elements with performance issues (red/yellow links and tiers).
- View network metric charts in tier/link popups.
- Find root causes in a specific network link: right-click and choose View Metrics. This opens a context-sensitive dashboard for the element.
  - The tier dashboard shows correlations between outlier (Slow/Very Slow/Stalled/Error) transactions and network-performance metrics.
  - The link dashboard shows network-performance issues on the client tier, server tier, and network path.
- Find root causes in a specific connection: when you narrow the issue down to a specific network link, you can
  - Configure the Network Agents to collect detailed metrics for the individual Connections used on that link.
  - Drill down to find the root cause in an individual Connection.
- You can copy IPs and ports for individual connections from the Network Dashboard. This makes it easy to forward this information to operations and other personnel when troubleshooting an issue. To copy IPs and ports:
  1. Click on a link to open the link popup and go to the Connections tab.
  2. Select the connections of interest (use ctrl-click to select multiple connections).
  3. Right-click on the selection and choose Copy IP addresses to clipboard.

### Important Notes

Note the following:
In many cases, the Application Dashboard and Network Dashboard might show different topologies for the same application. This is the expected behavior. Unlike app server agents, network agents detect intermediate network devices between nodes and consider these devices when collecting metrics. The two dashboards show the same application but from different perspectives: an application perspective and a multi-layer, network/application perspective.

This release does not support the following Network Flow Map features:
- Federated Flow Maps
- Visualization of flows between web servers and APM entry tiers
- The Network Flow Map does not filter out connections based on the selected time range.

In some cases, the KPIs for a tier or link might be different in the Network tab vs. the Flow Map. The popup window shows the latest data; the Flow Map updates data every two minutes. Any discrepancy between KPI values is due to this difference in reporting times.

If you open the Network Flow Map for an individual node, the KPI metrics for node-to-load-balancer and node-to-tcp-endpoint links show KPIs for all nodes in the parent tier (instead of KPIs the individual node only). To view KPIs for the individual node, open the link popup and look at the Connection KPIs.

Network Dashboard Reference

The Network Dashboard shows:

- Network-based key performance indicators (KPIs) 1 for the monitored nodes in each tier:
  - The average latency (μs) for individual packets
  - The number of Performance Impacting Events (PIE)
  - The number of errors

- KPIs for all network links 2. Click a tier or link to see the set of all associated Connections. (A gray link indicates a Connection that a network agent detected but did not directly monitor.)
The Network Dashboard shows a TCP endpoint icon when a network agent observes traffic between a monitored node and an endpoint (IP address & TCP port & protocol), but cannot determine if the endpoint represents:
- An application node that sends/receives traffic for the application, or
- An intermediate device (proxy or load balancer) that transfers traffic to/from another node over a separate Connection.

Network agents automatically detect load balancers, which often mask internal IP addresses and have the effect of splitting an in-flight application message into two separate TCP connections. The auto-detection of TCP endpoints and load balancers between application nodes make it easy to identify the exact locations where network issues are occurring.

Note how links are colored in the Network Dashboard:
- Green/yellow/red, to visualize the network performance compared to the overall performance baseline for that link.
- Blue, if baselining is disabled.
- Gray, for connections between application nodes and TCP endpoints. In this case, the gray link means "The connected node and endpoint are either the same device, or two devices exchanging traffic over a separate, unmonitored Connection."

The Network Dashboard also shows KPIs for the entire application. Here you can see the overall throughput that the application places on the network; the rate of network packet loss (packets sent but never received); the number of Performance Impacting Events (PIE), which indicate potential problems on one or more nodes or Connections; and the number of errors for the entire application.
Connection Explorer

The Connection Explorer (Explorer link, right side of Network Dashboard) shows all application flows and TCP Connections for an application in one table. The Connection Explorer is especially useful for large, complex applications. You can sort the table based on network KPIs to find flows with performance issues; then you can drill down and analyze the supporting connections in right-click Connection Dashboards.

The Connection Explorer shows information for:

- Application Flows (Top-Level Tree Nodes)
- TCP Connections (Second-Level Tree Nodes)

Application Flows (Top-Level Tree Nodes)

The Explorer shows the following information for each flow. Click View Options to show/hide information as appropriate.

- Source tier
- Destination tier
- Network Element

Tip
If two flows connect via a Load Balancer (LB), the table shows the LB IP:port and the connection source:

Flow direction: LB --> Tier

Flow direction: Tier --> LB

If you want to see the relationships between application tiers and intermediate Load Balancers (LBs) more clearly, you can drag the Network Element heading so that the columns are ordered Source | Network Element | Destination.
- Number of Connections used by the flow
- Latency
- PIE
- TCP Loss (Mille)
- Errors

TCP Connections (Second-Level Tree Nodes)

The Explorer shows the following information for each connection.

- Source IP, or Fully Qualified Domain Name (when available) for Load Balancers and TCP Endpoints
- Destination IP, or Fully Qualified Domain Name (when available) for Load Balancers and TCP Endpoints
**Custom Dashboards for Network Visibility**

You can create Custom Dashboards based on Network Visibility metrics using the standard workflows. To add a Network Visibility metric to a custom dashboard, do the following:

1. In the **Add Widget** window, click `+` (under **Data**).
2. Under **Data Source**, select the application of interest.
3. Under **Select a Metric Category**, choose **Custom (use any metrics)**.
4. Under Select a Metric, navigate to the metric of interest under one of the following:
   - Application Infrastructure Performance > Advanced Network
   - Application Infrastructure Performance > `<tier-name>` > Advanced Network
Network Visibility Events

On this page:
- Dynamic Monitoring Mode Events
- State-Change Events
- Start Events

Related pages:
- FAQs for Network Visibility

Packet Capture Events
Sub-type: PCAP (set by network agent)

<table>
<thead>
<tr>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet capture successful, Packet capture ID: &lt;id&gt;, Captured file: &lt;file.pcap&gt;</td>
<td>Info</td>
</tr>
<tr>
<td>Packet capture already active, Packet capture ID: &lt;id&gt;</td>
<td>Info</td>
</tr>
<tr>
<td>Packet capture couldn't be started, Storage location &lt;/loc/path&gt; doesn't exist, Packet capture ID: &lt;id&gt;</td>
<td>Err</td>
</tr>
<tr>
<td>Not enough space to start packet capture, Packet capture ID: &lt;id&gt;</td>
<td>Err</td>
</tr>
<tr>
<td>Packet capture failed. Capture directory &lt;/loc/path&gt; doesn't exist, Packet capture ID: &lt;id&gt;</td>
<td>Err</td>
</tr>
<tr>
<td>Packet capture successful and exported, Packet capture ID: &lt;id&gt;, Captured file: &lt;file.pcap&gt;</td>
<td>Info</td>
</tr>
<tr>
<td>Packet capture successful but export failed, Packet capture ID: &lt;id&gt;, Captured file: &lt;file.pcap&gt;</td>
<td>Err</td>
</tr>
<tr>
<td>Packet capture not started. Agent Disabled, Packet capture ID: &lt;id&gt;</td>
<td>Err</td>
</tr>
</tbody>
</table>

Dynamic Monitoring Mode Events
Sub-type: Config (set by network agent)

<table>
<thead>
<tr>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Visibility Agent mode change successful, Last DMM: &lt;Kpi/Diagnostic/Advanced&gt;, Current DMM: &lt;Kpi/Diagnostic/Advanced&gt;</td>
<td>Info</td>
</tr>
<tr>
<td>Network Visibility Agent mode change failed, Current DMM: &lt;Kpi/Diagnostic/Advanced&gt;</td>
<td>Err</td>
</tr>
</tbody>
</table>

State-Change Events
Sub-type: Config (set by network agent)

<table>
<thead>
<tr>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Visibility Agent state change successful, Current state: &lt;Enabled/Disabled&gt;</td>
<td>Info</td>
</tr>
<tr>
<td>No change in Network Visibility Agent state, Current state: &lt;Enabled/Disabled&gt;</td>
<td>Info</td>
</tr>
<tr>
<td>Network Visibility Agent state change failed as Packet capture active, Current state: &lt;Enabled/Disabled&gt;</td>
<td>Err</td>
</tr>
<tr>
<td>Network Visibility Agent state change failed, Current state: &lt;Enabled/Disabled&gt;</td>
<td>Err</td>
</tr>
</tbody>
</table>
Network Visibility Agent started successfully, Current state: <Enabled/Disabled>

Start Events

<table>
<thead>
<tr>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Visibility Agent registered successfully</td>
<td>Info</td>
</tr>
<tr>
<td>Network Visibility Agent registration failed due to an invalid ID</td>
<td>Err</td>
</tr>
<tr>
<td>Network Visibility Agent started successfully, PID: &lt;pid&gt;</td>
<td>Info</td>
</tr>
</tbody>
</table>
Advanced Operations

Network Visibility includes the following advanced operations:

- **Resolving Unmapped Connections** – In some cases, there is not enough context to map a TCP Connection to an application flow automatically. This section describes how you can resolve these connections manually.
- **Packet Captures** -- A packet capture is a snapshot of live network traffic. Packet captures are very useful for advanced, in-depth network diagnostics and troubleshooting.
Packet Captures

A packet capture is a snapshot of live network traffic. Packet captures are very useful for in-depth network diagnostics and troubleshooting. When you discover a network issue that affects your applications, you can capture traffic using the Network Visibility agents and send the resulting data to your network or ops team for further analysis.

Network agents save packet captures as pcap files. A wide variety of network analysis tools support pcap: Wireshark, tcpdump, Windump, and many others.

Packet captures are supported on Linux platforms only.

Restrict packet capture privileges to authorized users only
Packet capture files include "raw" application data that might contain sensitive information. Any user with Account Owner or Administrator privileges can perform packet captures. For this reason, these roles should be assigned to authorized users only. See Roles and Permissions.

Before Starting

You must do the following setup on each host before you can capture packets on that host.

Do the following:

- In the Controller, click the gear icon in the top right ( ) and choose AppDynamics Agents > Network Visibility Agents.
- Right-click on the agent to set up and choose View Packet Capture Configuration.
- Set the capture properties as follows.

Capture Settings

- **Duration (sec)** – This should be long enough to capture at least one Business Transaction over the link that you want to troubleshoot.
- **Size** – The maximum size for any single capture file.

**Packet Capture Filename Prefix** – You must specify a prefix. It is good practice to include the hostname or another string that clearly identifies the node. The resulting pcap filename includes the prefix, the IP, the interfaces captured, and the timestamp.

If you specify a prefix of DataCenterNYC--., for example, the resulting pcap will have a filename such as DataCenterNYC--ip-10-0-21-101_any_1_2017_09_28_17_58_03.pcap

Storage Settings

- **Path** – If remote storage is disabled, the agent stores capture files in this folder on the agent host.
  - The specified folder must exist on the agent host. The default path is /opt/appdynamics/netviz/pcap.
  - The <network-agent-user> account (the one used to install and run the Network Agent) should have read and write permissions to this folder.
- **Maximum Allocated Space** – Maximum storage allotted for all capture files. This setting applies to both the agent host and the remote server. As new capture files are created, the agent deletes older files to free up space.

Storage Settings (SCP Server)
Remote Storage (upload to SCP Server) – With this option enabled, the agent uploads the capture file to the specified server when the packet-capture operation ends.

Host/Port/Username/Password/Path

- The local path must be defined on the remote server.
- The specified user account must have write permissions on the specified path.

Best Practices for Packet Captures

Packet Capture files can get very large very quickly. When a capture job is in progress, the network agent captures all bytes in all packets on all network interfaces that it monitors. The size of the capture file depends on the capture duration and the rate of packets sent and received on the network interfaces of the node. The duration should be long enough to capture a few Business Transaction calls between the two nodes, but no longer.

If you want to retain any capture file for archiving or extended analysis, copy the file from the storage folder as soon as the capture ends. This ensures that it does not get overwritten by newer files.

Packet capture operations generate a number of Network Visibility Events that you can use for monitoring and troubleshooting.

Creating a Packet Capture

Do the following:

- 1 Determine the Nodes to Capture
- 2 Start the Packet Capture

Determine the Nodes to Capture

Do the following:

- Go to the Network Dashboard, set the reporting duration to the last 5 minutes, and verify that the network issue you need to troubleshoot is currently active.
- Note the node(s) where you need to capture packets.
  - To troubleshoot a node, capture on the node (A).
  - To troubleshoot a link, capture on the two connecting nodes on each side of the link (B).
  - If the link is bisected by a load balancer, capture on both sides of the load balancer (C).

Start the Packet Capture

When you start a capture, the agent captures all packets sent and received by all network interfaces on the agent host. When the agent stops capturing (as specified by the Duration (sec) capture setting), it saves the pcap file in the folder specified by Storage settings.

There are two ways to start a capture:

- From the Agents Page
From the Node Dashboard

From the Agents Page

You can use this method to capture on one or more nodes:

1. In the Controller, click the gear icon in the top right (⚙️) and choose AppDynamics Agents > Network Visibility Agents.
2. Select the agents on the nodes where you want to capture. Use Ctrl-click or Shift-click to select multiple agents.
3. Right-click on a selected agent and choose Start Packet Capture.

From the Node Dashboard

You can also use this method to capture on a single node:

1. Drill down to the node in the Network Browser:
   a. Go to Tiers & Nodes, right-click on the node, and choose View.
   b. When the Node view appears, go to the Network Browser.
2. Right-click on the node and choose Start Packet Capture.
Resolving Unmapped Connections

After you set up the App Server Agent and the Network Agent together, the Network Agent does the following automatically:

- Monitors TCP connections,
- Identifies the application flow that uses each connection, and
- Maps the connection to the flow automatically.

In some cases, however, there is not enough context to associate a connection with a specific application flow. This is an unmapped connection. In this case, you need to map the connection manually. If a connection remains unmapped, the network agent collects metrics for that connection but cannot associate these metrics with any monitored application flow.

To resolve an unmapped connection, do the following:

1. Go to the Network Dashboard and click **Mapping Wizard** (under Connections on the right). You can also open this wizard directly from a Tier Dashboard if that tier has any unmapped connections.
2. The Connection Mapping Wizard appears. The left pane shows all tiers with unmapped connections. When you select a tier, the Connections list (right pane) shows all the unmapped connections for that tier. The workflow is to iterate through each tier and specify the mapping for each connection for that tier.

   For each unmapped connection, do the following:
   a. Determine the application flow that uses the unmapped connection.
   b. Select a tier in the left pane.
   c. In the right pane, group the IPs by Destination VIP or Prefix (bit mask).
   d. Right-click on a connection and choose Map Connection(s).
   e. Specify the **Call Type**, **Source Tier**, and **Destination Tier** for the application flow that uses that connection.

You need to know the source tier, destination tier, and the call type (ENTRY/EXIT) for the application flow. If you do not know this information, contact a developer or someone else who knows the application well.
Troubleshooting Network Visibility Problems

How do I troubleshoot the Network Agent?

- Network Agent Logs
- Network Agent Health Rule

Network Agent Logs

If you notice any performance issues related to the agent, do the following:

1. Stop the network agent:
   a. For ZIP packages:
      i. Identify the agent process: `ps ef | grep \ appd-netagent`
         The output provides the process ID (PID) of the Network Agent process.
      ii. `kill <network_agent_PID>`
   b. For RPM and DEB packages: `sudo service appd-netviz stop`
2. Check the following log files under `<network-agent-install>/logs`:
   a. appd-netmon.log
   b. appd-netagent.log.log

Network Agent Health Rule

The following default Health Rule is also useful for troubleshooting Network Agents:

```
Network-Host: Packet drops too high
```

This rule triggers an alert when packets get dropped between a Network Agent and the host interface. A high rate of packet drops on the host can result in inaccurate metrics.

If the issue persists, run the `collect.sh` script available in the `bin/` folder in the `agent` directory and share the logs with your support contact to expedite troubleshooting. This script is available for Network Agent version 4.5.9 or above only.
Monitoring Containers

You can use the Machine Agent to monitor application nodes running inside Docker containers and to identify container issues that impact application performance. By viewing and comparing APM metrics with the underlying container and server/machine metrics, you can easily answer the question: Is my application problem purely an application problem, or is the root cause in the container or the server?

The Machine Agent should be deployed inside a Docker container. The Machine Agent collects metrics for Docker containers on the same host, in addition to server and machine metrics for the host itself. The Controller shows all monitored containers, for each host, and the container and host IDs for each container.

In a BRIDGE networking mode, the containers will take on the container ID as the host name. If networking is in host mode, then the containers take on the node name of the host ID. This means every container on that node will have the same host ID. In this case you need to use the unique host ID settings. See Register the Container ID as the Host ID.

By default, the Machine Agent only monitors containers that have a running APM Agent, but this can be changed by setting the `sim.docker.monitorAPMContainersOnly` property on the Controller. See Controller Settings for Server Visibility.

To deploy a Machine Agent on a host outside a Docker container, create a symbolic link: `ln -s / /hostroot` on the host. This symbolic link enables the Machine Agent to collect host metrics along with Docker container metrics. When a Machine Agent is deployed inside a Docker container for monitoring, the symbolic link is automatically created when the volume mounts. If you want to grant more restrictive permissions you can use the following command to create symbolic links: `ln -s /proc /hostroot/proc; ln -s /sys /hostroot/sys; ln -s /etc /hostroot/etc`. These links can be made read-only, the AppDynamics agent does not need write privileges to these directories.

The following diagram illustrates how to deploy container monitoring:

- Install the Machine Agent (1) in a standalone container. The Machine Agent collects hardware metrics for each monitored container, as well as Machine and Server metrics for the host (3), and forwards the metrics to the Controller.
- The Machine Agent is able to monitor all containers that are running on that host, subject to the limits outlined below, and will report runtime metrics and metadata for every container. In addition, if any of the containers have an APM Agent installed (2), then the Machine Agent also correlates the container metadata and runtime metrics with the associated APM Node.

Container monitoring requires a Server Visibility license and version 4.3.3 or higher of both the Controller and the Machine Agent.
Before You Start

Note the following:

1. AppDynamics recommends that you use Docker CE/EE v17.03 or Docker Engine v1.13 with this product. Some data might be missing if you are using previous versions of Docker.
2. Container Monitoring is not supported on Docker for Windows or Docker for Mac.
3. AppDynamics strongly recommends that you follow these guidelines, Instrument JVMs in a Dynamic Environment when monitoring container-based applications.
4. The Machine Agent can monitor up to 120 running containers per host. The default maximum number of containers per host is 100. With the cgroup enabled flag set to true, the Machine Agent can monitor up to 600 containers per host. See Configuring Docker Visibility.
5. The Machine Agent collects metrics from containers with one or more running processes whose command line matches a configurable regex. By default, this regex matches all processes (*). You can override this behavior by editing the following regex in <machine_agent_home>/extensions/DockerMonitoring/conf/DockerMonitoring.yml:
   ```yaml
   containerMonitoringConfig.containerProcessSelectorRegex: ".*"
   ```
6. The maximum number of containers you can monitor in one Controller depends on the Controller size, the total number of App Agents, and the current load. If you are using a SaaS Controller, please work with your Account Manager.
7. Please review the following:
   a. Best Practices for setting up App Agents and Standalone Machine Agents
   b. Docker Visibility Known Issues in the latest Release Notes

Enabling Container Monitoring

To enable Container Monitoring:

1. On the Controller, log in to the Administration Console and verify that `sim.docker.enabled` is set to true.
2. On the agent, make sure that Server Visibility Enabled and Enable Docker Visibility are both set to true.
Container Monitoring Setup

The quickest and easiest way to run the Machine Agent with Container Monitoring enabled is to use one of the official images from the Docker Store: [https://store.docker.com/images/appdynamics](https://store.docker.com/images/appdynamics). These images are produced by AppDynamics, based on certified base images from the Docker Community, and these can either be run directly or used as base images for your own application containers. For full details of how to download and run containers based on these official images, please see the documentation posted on the Docker Store. If you prefer to build your base images, the full source code for building these images is posted to GitHub. You can use this as a pattern for your own builds: [https://github.com/Appdynamics/appdynamics-docker-images](https://github.com/Appdynamics/appdynamics-docker-images).

You should set the following for the Machine Agent to monitor containers running on the server. See Deploy Using Containers.

- **Server Visibility Enabled** – Enable Server Visibility
- **Docker Enabled** – Enable Docker Visibility
- **Volume Mounts** – Specify the following:
  - Volume mounts to allow read-only access to the underlying file system (/proc, /etc and /sys). This allows the Server Agent to collect host-level metrics for containers running on the server.
  - The UNIX domain socket on which the Docker daemon is configured to listen for API calls.

Viewing Container Details

To view container metadata and metrics in the Controller:

- In the Applications Dashboard, go to Containers to see all monitored containers used by the application.

- In the Servers Dashboard, go to Containers tab to see all monitored containers on that host.

- To open the Container Dashboard, right-click on the container name, and choose View Details.
The Container Details view has the following tabs, which provide an overview of the health and resource usage for the container:

a. **Overview** - Container metadata, Tags (name-value pairs derived from Docker/Kubernetes) plus AWS tags where applicable, and single chart views for CPU, memory, network and disk usage.
b. **CPU** - CPU Usage and Throttled Time metrics
c. **Memory** - Memory Usage and Memory Fault metrics
d. **Network** - Network Usage metrics (only available when running in Diagnostic mode). For more information on Diagnostic mode, see "Enable Dynamic Monitoring Mode (DMM)" under Standalone Machine Agent Configuration Properties.

- The Node Dashboard also includes a Containers tab for the container in which that node is running.

**Viewing Container Metrics using the Metric Browser**

To view time-series metric data for containers, double-click on one of container metric graphs (CPU, Memory, Network, Disk). This will open the Metric Browser with the displayed metric selected. The Metric Browser tree displays the full set of metrics available for that container, and you can add these to the Metric Browser display by double-clicking on the metric you wish to select. See Container Metrics.
Troubleshooting Containers using Dynamic Monitoring Mode

You can selectively control the number of Container metrics reported for individual containers, using Dynamic Monitoring Mode (DMM). Rather than have all agents report all metrics for all containers all the time, you can configure individual agents to collect:

- Key Performance Indicator metrics only (KPI mode)
- KPI and Diagnostic metrics (Diagnostic mode)
- All available metrics (Advanced Diagnostic mode)

This provides the flexibility to report KPI metrics only on most containers and then increase the metric level on specific containers where you need deeper visibility to diagnose problems. You can increase scalability on the Controller and conserve metric bandwidth on the network with no sacrifice in visibility. Every Docker Visibility metric has a default DMM class (KPI, Diagnostic, and Advanced Diagnostic). Consult the Container Metrics table for details: the Default Monitoring Mode column indicates the default category of each metric when Dynamic Monitoring Mode is enabled.

When you notice a container that requires additional analysis or troubleshooting, change the DMM on that container to collect more metrics:

1. Open the Container Detail window for the container of interest: Open the Containers table (Containers, left menu) and double-click on the container.
2. Click the Edit (pen) button in the top-right corner, change the mode to Diagnostic or Advanced Diagnostic, and click the checkmark to apply the changes.
3. Collect additional metrics for your analysis or troubleshooting.
4. When you no longer need to collect additional metrics for that container, set the DMM on that container back to KPI.

Note the following:

- The Controller has three different DMM defaults, depending on the type of metrics collected by the agent. These defaults are all set to KPI (the lowest setting).
- `sim.machines.dmm.containerAware.defaultMode`
  Default for **Hardware Resources Metrics** on servers with Docker Visibility enabled
- `sim.machines.dmm.container.defaultMode`
  Default for **Docker Container Metrics** on containers

- The default for both settings is KPI (the lowest setting).
- AppDynamics recommends that you leave these global settings at their defaults. If you need to collect advanced metrics for a specific server or container, increase the DMM on an "as-needed" basis as described in the following section.
- When a container monitored by a Java App Agent shuts down and restarts, any overridden DMM specified for the shut-down container is lost. The DMM for the restarted container automatically resets to the global default specified by `sim.machines.dmm.container.defaultMode`.
- If you switch the monitoring mode in the Controller from a more-inclusive to a less-inclusive mode, the Metric Browser will show values for the newly-excluded metrics for one hour after the switch. For example: suppose you switch from Diagnostic to KPI mode. For any Diagnostic metric, the Metric Browser will report a steady line (at 0 or the last-reported value) for one hour after the switch; then the line will disappear. This is standard behavior in the Metric Browser for an agent when it stops reporting a specific metric.
Monitoring Kubernetes

With AppDynamics you can gain real-time visibility into your containerized applications deployed to Kubernetes. Kubernetes is an open-source container-orchestration platform for automating deployment, scaling, and management of applications running in containers.

With container visibility, you can enhance container-level metrics, and gain visibility into CPU, packet visibility, memory, and network utilization. These metrics can then be baselined, and have health rules associated along with detailed resource usage statistics about your APM-monitored container applications. By viewing and comparing APM metrics, with the underlying container and server metrics, you quickly receive deep insights into the performance of your containerized applications, along with potential impediments in your infrastructure stack. For example, specific metrics can help you identify both "bandwidth-hogging" applications and container-level network errors.

Container visibility allows you to monitor containerized applications running inside Kubernetes pods and to identify container issues that impact application performance. The agent is deployed as a Kubernetes DaemonSet in every node of a Kubernetes cluster. Deploying the Machine Agent as a DaemonSet ensures that every Kubernetes worker node runs the Machine Agent and that the agent collects critical resource metrics from both the node host and the associated Docker containers.

Container Visibility with Kubernetes

Deploy the Machine Agent in Docker-enabled mode. For more information and details on how to configure and run the Machine Agent using Docker, see Configuring Docker Visibility. The Machine Agent will then:

- Identify the containers managed by Kubernetes.
- Determine if these containers contain App Server Agents.
- Correlate containers with App Server Agents with the APM nodes for that application.

The following diagram illustrates the deployment scenario for container visibility in Kubernetes:
Install the Machine Agent container as a DaemonSet on each Kubernetes node.

If you wish to collect APM metrics from any container in a pod, install the correct APM Agent in the container before deploying the pod.

The Machine Agent collects resource usage metrics for each monitored container, as well as Machine and Server metrics for the host, and forwards the metrics to the Controller.

(Optional) Install the Network Agent as a DaemonSet on the node you want to monitor. The Network Agent collects the metrics for all network connections between application components being monitored and sends the metrics to the Controller.

Before You Begin

Container visibility with Kubernetes requires the following:

- The Machine Agent must run as a DaemonSet on every Kubernetes node that you wish to monitor.
- Each node to be monitored must have a Server Visibility license.
- Docker Visibility must be enabled on the Machine Agent.
- Both App Server Agents and Machine Agents are registered by the same account and are using the same Controller.
- If you have multiple App Server agents running in the same pod, register the container ID as the host ID on both the App Server Agent and the Machine Agent.

Limitations
Only the Docker Container Runtime is supported.
Only Pod and ReplicaSet labels are supported.

**Enable Container Visibility**

Update the Controller to 4.4.3 or higher if you have not already done so. To enable Kubernetes visibility in your environment, edit the following parameters:

**Controller**

- `sim.machines.tags.k8s.enabled`: The value defaults to true. The global tags enabled flag has priority over this.
- `sim.machines.tags.k8s.pollingInterval`: The value defaults to one minute. The minimum value you can set for the polling interval is 30 seconds.

**Machine Agent**

- `k8sTagsEnabled`: The value defaults to true and is specified in the `ServerMonitoring.yml` file.

Continue with Monitoring Red Hat OpenShift. You can use the example DaemonSet, the sample Docker image for Machine Agent, and the sample Docker start script to quickly set up the Standalone Machine Agent.

**Register the Container ID as the Host ID**

Install an App Server Agent in each container in a Kubernetes pod to collect application metrics. If multiple App Server agents are running in the same pod, in the Redhat OpenShift platform for example, you must register the container ID as the unique host ID on both the App Server Agent and the Machine Agent to collect container-specific metrics from the pod. Kubernetes pods can contain multiple containers and they share the same host ID. The Machine Agent cannot identify different containers running in a pod unless each container ID is registered as the host ID.

To register the container ID as the host ID:

1. Get the container ID from the cgroup:

   ```
   cat /proc/self/cgroup | awk -F '/' '{print $NF}' | head -n 1
   ```

2. Register the app server agents:

   ```
   -Dappdynamics.agent.uniqueHostId=$(sed -rn '1s#.*/##; 1s/(.{12}).*/\1/p' /proc/self/cgroup)
   ```

   For OpenShift, run the following command:
   ```
   -Dappdynamics.agent.uniqueHostId=$(sed -rn '1s#.*/##; 1s/docker-(.{12}).*/\1/p' /proc/self/cgroup)
   ```

3. Register the Machine Agent:
Instrument Applications with Kubernetes

There are several approaches to instrumenting applications deployed with Kubernetes, and which one you choose will depend on your particular requirements and DevOps processes. In order to monitor an application container with AppDynamics, an APM Agent must be included in that container. This can be done in a number of ways:

1. Using an appropriate base image which has the APM agent pre-installed
2. Loading the agent dynamically as part of the container startup
3. Loading the agent and dynamically attaching to a running process (where the language runtime supports it)

Option 3 is usually applicable only to Java-based applications since the JVM supports Dynamic Attach, which is a standard feature of the AppDynamics Java APM Agent. See this blog for more details and an example of how to do this. For the other options, it is common practice to make use of standard Kubernetes features such as Init Containers, ConfigMaps, and Secrets as described in this blog.

Deploy the Machine Agent on Kubernetes

AppDynamics Machine Agent can be deployed in a single container image, without the need for an init container. By default the machine agent is deployed to the cluster as a DaemonSet, to distribute each agent instance evenly across all cluster nodes. Where required, the daemon set can be configured with node affinity rules or node anti-affinity rules to ensure that it is deployed to a desired set of nodes and not across the entire cluster. There is more information on node affinity here.

In order to harvest pod metadata, the service account used to deploy the machine agent must have the **cluster-reader** role in OpenShift. The "cluster-reader" role is also required for the Kubernetes extensions to the machine agent.

```bash
# assigning cluster-reader role in OpenShift
oc adm policy add-cluster-role-to-user cluster-reader -z appd-account
```

If you are working with a vanilla Kubernetes distribution, it may not have a pre-built cluster role similar to "cluster-reader" in OpenShift. Please see ClusterRole Configuration for details of how to do this.

Resource Limits

- The main application being monitored should have resource limits defined. Provide 2% padding for CPU and add up to 100 Mb of memory.
- To support up to 500 containers, the Machine Agent can be configured with the following resource requests and limits: Mem = 400M, CPU = "0.1" and limits: Mem = 600M, CPU = "0.2"

AppDynamics provides a Kubernetes Snapshot Extension for monitoring the health of the Kubernetes Cluster. When deploying this extension, it is important to keep in mind that only a single version of the extension should be deployed to the
ClusterRole Configuration

Below is a sample role definition that provides a wide read access to various Kubernetes resources. These permissions are more than sufficient to enable Kubernetes extensions to the machine agent as well as pod metadata collection. The role is called 'appd-cluster-reader', but you can obviously name it as necessary. The cluster role definition outlines various api groups that will be available for members of this role. For each api group, we define a list of resources that will be accessed and the access method. Because we only need to retrieve information from these api endpoints, we only need the read-only access, expressed by "get", "list" and "watch" verbs.

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: appd-cluster-reader
rules:
- nonResourceURLs:
  - '*'
  verbs:
    - get
- apiGroups: ["batch"]
  resources:
    - "jobs"
  verbs: ["get", "list", "watch"]
- apiGroups: ["extensions"]
  resources:
    - daemonsets
    - daemonsets/status
    - deployments
    - deployments/scale
    - deployments/status
    - horizontalpodautoscalers
    - horizontalpodautoscalers/status
    - ingresses
    - ingresses/status
    - jobs
    - jobs/status
    - networkpolicies
    - podsecuritypolicies
    - replicasets
    - replicasets/scale
    - replicasets/status
    - replicationcontrollers
    - replicationcontrollers/scale
```

Do not include it in the DaemonSet to avoid duplicates and potential cluster overload. Instead, consider deploying the instance of the Machine Agent with the extension as a separate deployment with 1 replica in addition to the daemon set for Server Visibility. The machine agent SIM and Docker can be disabled in this case and the memory request can be dropped to 250M.
- storageclasses
- thirdpartyresources
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources:
  - bindings
  - componentstatuses
  - configmaps
  - endpoints
  - events
  - limitranges
  - namespaces
  - namespaces/status
  - nodes
  - nodes/status
  - persistentvolumeclaims
  - persistentvolumeclaims/status
  - persistentvolumes
  - persistentvolumes/status
  - pods
  - pods/binding
  - pods/eviction
  - pods/log
  - pods/status
  - podtemplates
  - replicationcontrollers
  - replicationcontrollers/scale
  - replicationcontrollers/status
  - resourcequotas
  - resourcequotas/status
  - securitycontextconstraints
  - serviceaccounts
  - services
  - services/status
  verbs: ["get", "list", "watch"]
- apiGroups:
  - apps
  resources:
  - controllerrevisions
  - daemonsets
  - daemonsets/status
  - deployments
  - deployments/scale
  - deployments/status
  - replicaset
  - replicaset/scale
  - replicaset/status
  - statefulsets
  - statefulsets/scale
  - statefulsets/status
verbs:
- get
- list
- watch
- apiGroups:
  - apiextensions.k8s.io
    resources:
    - customresourcedefinitions
    - customresourcedefinitions/status
    verbs:
    - get
    - list
    - watch
- apiGroups:
  - apiregistration.k8s.io
    resources:
    - apiservices
    - apiservices/status
    verbs:
    - get
    - list
    - watch
- apiGroups:
  - events.k8s.io
    resources:
    - events
    verbs:
Once the role is defined, you will need to create cluster role bindings to associate the role with a service account. Below is an example of a ClusterRoleBinding spec that makes appd-cluster-reader service account a member of the appd-cluster-reader-role in project “myproject”. Note that the naming is purely coincidental. The names of the service account and the cluster role do not have to match.

A Sample ClusterRoleBinding

```yaml
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cluster-reader-role-binding
subjects:
- kind: ServiceAccount
  name: appd-cluster-reader
  namespace: myproject
roleRef:
  kind: ClusterRole
  name: appd-cluster-reader
  apiGroup: rbac.authorization.k8s.io
```
Monitoring Red Hat OpenShift

Kubernetes Visibility extends its support to OpenShift Version 3. With this support, you can collect performance data from OpenShift clusters by using the Machine Agent. You install the agent as a DaemonSet on each host in the OpenShift cluster. The DaemonSet monitors each host on the OpenShift cluster and the corresponding containers that have an AppDynamics App Agent running. The process is similar to that for monitoring Kubernetes cluster.

The topic provides instructions on how to set up the Machine Agent running as a DaemonSet on an OpenShift cluster. The process includes the following:

- Creating the machine-agent project
- Creating a service account for running the Machine Agent
- Building a Docker image of the Machine Agent
- Deploying the Machine Agent

Before You Start

- This functionality requires a Server Visibility license.
- AppDynamics recommends that you use Docker CE/EE v17.03 or Docker Engine v1.13 with this product. Some data might be missing if you are using previous versions of Docker.
- Make sure you have an OpenShift user account with cluster-admin role, such as system:admin. This user is responsible for configuring the Machine Agent.
- Install the OpenShift command-line tool, oc.

Setting Java Options

The metaspace max is set to 100mb by default in OpenShift. If your application operates within a small margin of its existing memory resource allocation, you may choose to increase the allocation for the application. AppDynamics recommends allocating additional Metaspace Size space to accommodate the machine agent. Configure the environment variable GC_MAX_METASPACE_SIZE to set the MaxMetaspaceSize parameter. This can be done directly in the deployment, through a config map.

Create Project and Service Account

In OpenShift terminology, a project is a mechanism to isolate a group of users and their resources. An administrator can provide individual users or groups with permissions to create projects or manage specific projects. To isolate the Machine Agent from other projects, you can create a project for it.

To do so, create the machine-agent project:

```
oc new-project machine-agent
```

Create a service account, ma, with necessary privileges for the Machine Agent to obtain metrics:

1. Check whether the current project is machine-agent:

```
oc status
```

2. Create a service account.
Service accounts provide a secure way to control OpenShift API access without sharing a regular user’s credentials.

3. **Assign the privileged security context constraints (SCC) to the service account.**
   SCC determines the permissions and abilities of pods. See Security Permissions for the Service Account for the detailed list of permissions for the service account.

   ```
   oc create serviceaccount ma
   ```

   ```
   oc adm policy add-scc-to-user privileged -z ma
   ```

4. Add cluster-reader to the service account.

   ```
   oc adm policy add-cluster-role-to-user cluster-reader -z ma
   ```

### Security Permissions for the Service Account

Service account requires the following security permissions to provide maximum isolation to the project under which the Machine Agent is running:

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The cluster-reader role</td>
<td>This cluster-reader role allows the Machine Agent to read tags from the OpenShift cluster. An example:</td>
</tr>
<tr>
<td></td>
<td><code>oc adm policy add-cluster-role-to-user cluster-reader -z ma</code></td>
</tr>
<tr>
<td>The privileged SCC</td>
<td>The privileged SCC allows the Machine Agent to be run as the root user.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td><code>oc adm policy add-scc-to-user privileged -z ma</code></td>
</tr>
<tr>
<td>Run as privileged container</td>
<td>Running as the privileged container allows the Machine Agent to perform operations such as:</td>
</tr>
<tr>
<td></td>
<td>- Reading files from /etc</td>
</tr>
<tr>
<td></td>
<td>- Reading process information from /proc</td>
</tr>
<tr>
<td></td>
<td>This privilege is configured in the DaemonSet YAML file as follows:</td>
</tr>
<tr>
<td></td>
<td><code>securityContext:</code></td>
</tr>
<tr>
<td></td>
<td><code>privileged: true</code></td>
</tr>
</tbody>
</table>

### Creating Docker Image of the Machine Agent

Copy the following files to a directory on a machine that can build the Docker image:

- Machine Agent Bundle - 64-bit Linux (zip)
  Download this bundle and rename it to `machine-agent.zip`. 
**Deploy Machine Agent as DaemonSet**

1. Modify the sample DaemonSet to point to your Controller. The major sections are highlighted below:
   a. Select the worker node:

   ```yaml
   nodeSelector:
     node-role.kubernetes.io/compute: "true"
   ```

   b. Change the following Controller information in the sample DaemonSet:

   ```yaml
   containers:
   - env:
     - name: APPDYNAMICS_CONTROLLER_HOST_NAME
       value: "<controller-host-name>"
     - name: APPDYNAMICS_CONTROLLER_PORT
       value: "<controller-port>"
     - name: APPDYNAMICS_CONTROLLER_SSL_ENABLED
       value: "true"
     - name: APPDYNAMICS_AGENT_ACCOUNT_ACCESS_KEY
       value: "<account-access-key>"
     - name: APPDYNAMICS_AGENT_ACCOUNT_NAME
       value: "<your-account>"
     - name: APPDYNAMICS_SIM_ENABLED
       value: "true"
     - name: APPDYNAMICS_DOCKER_ENABLED
       value: "true"
   ```

   c. Specify the Docker image:

   ```yaml
   image: "<your image>"
   ```

   d. Enable the Machine Agent to run as the privileged user:
1. If you have a different service account name, replace it with the existing one:

   ```yaml
   serviceAccount: ma
   serviceAccountName: ma
   ```

e. If you have a different service account name, replace it with the existing one:

   ```yaml
   securityContext:
     privileged: true
   ```

2. Create DaemonSet below:
   a. Check whether the current project is machine-agent:

   ```bash
   oc status
   ```

   b. Create the machine-agent DaemonSet:

   ```bash
   oc create -f machine-agent-daemonset.yaml
   ```

   The Machine Agent appears in the UI.

**Check Status**

If the Machine agent does not appear in the UI, check the status as follows:

1. Check the machine-agent DaemonSet.

   ```bash
   oc get ds
   ```

2. Check the machine-agent pod.

   ```bash
   oc get pod
   ```
The pod name is machine-agent-daemonset--<XXXX>.

3. Check logs whether the last line is ”Started AppDynamics Machine Agent Successfully”.

```
oc get logs -f <machine-agent-daemonset--<XXXX>>
```

Sample DaemonSet

Modify the sample DaemonSet to suit your deployment scenario.

```yaml
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: machine-agent-daemonset
  namespace: machine-agent
spec:
  selector:
    matchLabels:
      name: appdynamics-machine-agent
  template:
    metadata:
      labels:
        name: appdynamics-machine-agent
    spec:
      nodeSelector:
        node-role.kubernetes.io/compute: "true"
      containers:
      - env:
          - name: APPDYNAMICS_CONTROLLER_HOST_NAME
            value: "<your-hostname>"
          - name: APPDYNAMICS_CONTROLLER_PORT
            value: "<your-port>"
          - name: APPDYNAMICS_CONTROLLER_SSL_ENABLED
            value: "true"
          - name: APPDYNAMICS_AGENT_ACCOUNT_ACCESS_KEY
            value: "<your-access-key>"
          - name: APPDYNAMICS_AGENT_ACCOUNT_NAME
            value: "<your-account>"
          - name: APPDYNAMICS_SIM_ENABLED
            value: "true"
          - name: APPDYNAMICS_DOCKER_ENABLED
            value: "true"
        image: <your-image>
        name: machine-agent
        securityContext:
          privileged: true
        volumeMounts:
```
- mountPath: /hostroot
  name: hostroot
  readOnly: true
- mountPath: /var/run/docker.sock
  name: docker-sock
restartPolicy: Always
imagePullPolicy: Always
serviceAccount: ma
serviceAccountName: ma
volumes:
- name: hostroot
  hostPath:
    path: /
- name: docker-sock
  hostPath:
    path: /var/run/docker.sock

Sample Dockerfile

You can either use this example Dockerfile or the one given in OpenShift Visibility Manifests.

```docker
# Sample Dockerfile for the AppDynamics Machine Agent
# This is provided for illustration purposes only

FROM ubuntu:16.04
# Install required packages
RUN apt-get update && \
    apt-get install -y unzip && \
    apt-get clean
# Install AppDynamics Machine Agent
ENV MACHINE_AGENT_HOME /opt/appdynamics/machine-agent/
ADD machine-agent.zip /tmp/machine-agent.zip
RUN mkdir -p ${MACHINE_AGENT_HOME}
# Include start script to configure and start MA at runtime
ADD start-appdynamics ${MACHINE_AGENT_HOME}
RUN chmod 774 ${MACHINE_AGENT_HOME}/start-appdynamics
# change files and directories to be owned by root so MA container
user in root
group can access
RUN chgrp -R 0 /opt && \
    chmod -R g=u /opt
RUN chgrp -R 0 /tmp/machine-agent.zip && \
    chmod g=u /tmp/machine-agent.zip
# Changing directory to MACHINE AGENT HOME
WORKDIR ${MACHINE_AGENT_HOME}
# Configure and Run AppDynamics Machine Agent
CMD "./start-appdynamics"
```

Sample start-appdynamics Script

- `chgrp` is used to change the group ownership of directories and files.
  - `-R` flag recursively changes the group ownership.
  - `g=u` changes the group ownership to the same as the user.
#!/bin/bash
unzip -oq /tmp/machine-agent.zip -d ${MACHINE_AGENT_HOME} && \
   rm /tmp/machine-agent.zip
# Start Machine Agent
./bin/machine-agent -j jre/

Security Considerations

You can deploy the Machine Agent with tighter security. Determine the level of security for the project and leverage the following permissions to secure the Machine Agent.

Run the Machine Agent Without cluster-reader Role

Without the cluster-reader role, the Machine Agent cannot read information, such as Pod and ReplicaSet, from the OpenShift cluster. However, it can collect other metrics except for the tags for the app server agent container.

Run the Machine Agent Without Privileged Container Mode

To turn off this privilege, remove the section below from the DaemonSet YAML file.

```yaml
securityContext:
  privileged: true
```

Without the Privileged Container mode, the Machine Agent cannot read files, such as `/hostroot/etc/passwd` and `/hostroot/proc/<pid>/etc`, and therefore it cannot collect metrics such as:

- Processes
- Network

Turning off this privilege has no effect on collecting the app agent container metrics.
Monitoring Pivotal Cloud Foundry

Related pages:

- Where can I find information on the AppDynamics integration with Pivotal Cloud Foundry (PCF)
- AppDynamics Application Performance Monitoring for PCF [Pivotal Network]
- AppDynamics Platform Monitoring for PCF [Pivotal Network]
- AppDynamics Application Analytics for PCF [Pivotal Network]
- Deploying Machine Agent on PCF
- FAQs and Troubleshooting Guide

PCF

Pivotal Cloud Foundry (PCF) is an open-source cloud computing Platform as a Service (PaaS). Developers can develop, deploy, operate, and scale cloud-native applications for public and private clouds.

AppDynamics provides three tiles to support PCF 2.x (v2.0.7+) that are available on the Pivotal Network for download and install in a PCF foundation.

PKS

Pivotal Container Service or PKS is Pivotal’s product for deploying and managing open-source Kubernetes. The Kubernetes monitoring that AppDynamics provides has been tested to work with PKS. Follow the instructions from Monitoring Kubernetes for setting up container visibility within PKS. This daemonset.yaml example has been tested with kubectl in PKS. The yaml file contains a command that sets up a symbolic link. This symbolic link maps the location of the docker.sock file, expected by the AppDynamics Machine Agent, to the actual location on a PKS worker node.

AppDynamics Application Performance Monitoring for PCF

To simplify APM setup, the AppDynamics Application Performance Monitoring for PCF tile delivers a service broker to support an AppDynamics marketplace service. The Controller parameters are configured in the tile and automatically published to the marketplace when the tile is installed.

A PCF app can be bound to the marketplace service to consume the Controller credentials.

```
$ cf bind my-pcf-app my-appd-marketplace-service
```

Performing a push will utilize the integrated AppDynamics buildpack support to start the APM agent based on the parameters from the marketplace service and start sending performance metrics to the Controller.

```
$ cf push
```
The tile also installs the AppDynamics extension buildpack in the PCF environment. For more information on the buildpack support and language-specific workflows to enable APM, see the tile documentation.

**AppDynamics Platform Monitoring for PCF**

The AppDynamics Platform Monitoring for PCF tile provides visibility into the PCF platform where your AppDynamics-monitored applications are deployed. The tile deploys a Firehose Nozzle that collects various KPI metrics and publishes them to a Controller. The tile also deploys a Dashboard App that automatically creates KPI health rules and custom dashboards associated with the health rules. The Dashboard App allows you to monitor the health and capacity of one or more PCF foundations. For more information see the tile documentation.

**AppDynamics Application Analytics for PCF**

The AppDynamics Application Analytics for PCF tile supports collecting Transaction Analytics in real-time from PCF apps to connect application performance, user experience, and business outcomes. The tile deploys an analytics agent in the PCF environment that forwards transaction analytics events from PCF apps to an AppDynamics Events Service. For more information see the tile documentation.

**Deploy Using Pivotal Container Service (PKS)**

Pivotal Container Service or PKS is Pivotal’s product for deploying and managing open-source Kubernetes. The Kubernetes monitoring that AppDynamics provides has been tested to work with PKS. Follow the instructions from Monitoring Kubernetes for setting up container visibility within PKS.

Below is a daemonset.yaml example that has been tested with kubectl in PKS and contains a command to setup a symbolic link to map the location of the docker.sock file expected by the AppDynamics Machine Agent to the actual location on a PKS worker node.

**Sample Daemonset**

Here is an example YAML file to run Machine Agent as Daemonset across your Kubernetes nodes:

```yaml
---
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: sim-k8s-agent
  namespace: default
spec:
template:
  metadata:
    labels:
      name: appdynamics
  spec:
    containers:
    - env:
      - name: APPDYNAMICS_CONTROLLER_HOST_NAME
        value: "<controller-host-name>"
      - name: APPDYNAMICS_CONTROLLER_PORT
        value: "<controller-port>"
      - name: APPDYNAMICS_CONTROLLER_SSL_ENABLED
        value: "<true/false>"
      - name: APPDYNAMICS_AGENT_ACCOUNT_NAME
```
value: "<account-name>"
- name: APPDYNAMICS_AGENT_ACCOUNT_ACCESS_KEY
  value: "<access-key>"
- name: APPDYNAMICS_SIM_ENABLED
  value: "true"
- name: APPDYNAMICS_DOCKER_ENABLED
  value: "true"
image: <machine-agent-image>
name: sim-k8s-agent
volumeMounts:
- mountPath: /hostroot
  name: hostroot
  readOnly: true
- mountPath: /var/run/docker.sock
  name: docker-sock
volumes:
- name: hostroot
  hostPath:
    path: /
    type: Directory
- name: docker-sock
  hostPath:
    path: /var/run/docker.sock
    type: Socket
  restartPolicy: Always
---
kkind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: cluster-read-all
rules:
  - apiGroups:
    - ""
      - apps
      - autoscaling
      - batch
      - extensions
      - policy
      - rbac.authorization.k8s.io
    resources:
      # everything except secrets
      - componentstatuses
      - configmaps
      - daemonsets
      - deployments
      - events
      - endpoints
      - horizontalpodautoscalers
      - ingress
- jobs
- limitranges
- namespaces
- nodes
- pods
- persistentvolumes
- persistentvolumeclaims
- resourcequotas
- replicasets
- replicationcontrollers
- serviceaccounts
- services
verbs:
- get
- watch
- list
- nonResourceURLs: ["*"]
verbs:
- get
- watch
- list

---

kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: k8s-agent-readall-role-binding
subjects:
  - kind: ServiceAccount
    name: default
    namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: cluster-read-all

---
Container Metrics

To access: AppDynamics Home > Servers > Containers

The Containers Dashboard shows resource utilization metrics of your containerized applications and the underlying host machine. You can access the Containers Dashboard from the Servers tab in the Controller UI. The Containers page lists all monitored containers used by an application that is registered to the Controller.

## Container Metrics

### CPU

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Busy</td>
<td>Percentage of time the CPU was busy processing system or user requests from this container; this metric includes CPU Stolen time.</td>
<td>KPI</td>
</tr>
</tbody>
</table>
| %Busy Scaled| If the `<cpus>=<value>` option is set in the container, this metric measures the scaled value of the CPUs property. If this value is not set, this metric measures the scaled value of the Logical CPUs property. To enable CPU Scaling, do the following:  
  - Log in to the administration console: http://<controller host>:<port>/controller/admin.jsp  
  - Select the user account of interest.  
  - Click Add Property and add the following:  
    ```
    key = sim.docker.container.cpuScaling
    value = true
    ``` | KPI                     |
| Logical Cores| The number of logical cores on the host machine that are available to the container. | KPI                     |
| Shares      | The `-c or cpu-shares` setting used to start the container.                  | KPI                     |
| Throttle Count| Number of times the container CPU is throttled. Putting this in Advanced mode as the first metric to look for throttling would be Throttled Time and if Throttled time is less and still container is limited on computation then this metric can be looked at. | KPI                     |
| Throttled Time (ms) | Total amount of time the container CPU was throttled and did not get enough host CPU. High throttle times indicate that The `-c or cpu-shares` setting used to start the container was too low. | KPI                     |
| Total %     | The average utilization (all available CPUs) by the container.               | KPI                     |

### Disks
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average IO</td>
<td>Average disk utilization for read/write operations. This metric might be unavailable on some platforms.</td>
<td>KPI</td>
</tr>
<tr>
<td>Utilization %</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;disk&gt; Reads/sec</td>
<td>Number of read operations by the container on &lt;disk&gt;.</td>
<td>Diagnostic</td>
</tr>
<tr>
<td>&lt;disk&gt; Writes/sec</td>
<td>Number of write operations by the container on &lt;disk&gt;.</td>
<td>Diagnostic</td>
</tr>
<tr>
<td>KB Read/sec</td>
<td>The rate of data read by the container from all disks and partitions.</td>
<td>KPI</td>
</tr>
<tr>
<td>KB Written/sec</td>
<td>The rate of data written by the container to all disks and partitions.</td>
<td>KPI</td>
</tr>
<tr>
<td>MB Reads/sec (MB)</td>
<td>Number of read operations by the container on all disks and partitions.</td>
<td>KPI</td>
</tr>
<tr>
<td>MB Writes/sec</td>
<td>Number of write operations by the container on all disks and partitions.</td>
<td>KPI</td>
</tr>
<tr>
<td>Queue Size</td>
<td>Average number of container requests queued during the time bucket. Consistently high queue sizes indicate that the container is not getting enough IO resources. This metric might be unavailable on some platforms.</td>
<td>KPI</td>
</tr>
</tbody>
</table>

**Memory**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache Size (MB)</td>
<td>Total amount of data cached in memory.</td>
<td>Diagnostic</td>
</tr>
<tr>
<td>Failed Count</td>
<td>Number of times a memory allocation failure occurred because it exceeded the pre-set memory limit for that container. OR Number of times the hard limit for the container was hit.</td>
<td>KPI</td>
</tr>
<tr>
<td>Page Major Fault Count</td>
<td>Number of major page faults caused by the container. Docker metric pgmajfault. A high rate of page faults might indicate that the memory constraints specified for the container are too low.</td>
<td>KPI</td>
</tr>
<tr>
<td>Reservation</td>
<td>Soft limit based on the --memory-reservation used to start the container</td>
<td>KPI</td>
</tr>
<tr>
<td>Resident Set Size (MB)</td>
<td>Total amount of memory that belongs to stacks, heaps, and anonymous memory maps.</td>
<td>KPI</td>
</tr>
<tr>
<td>Usage</td>
<td>Memory usage by the container, in bytes.</td>
<td>KPI</td>
</tr>
<tr>
<td>Used %</td>
<td>Memory utilization by the container as a percentage of the memory hard limit, which is based on the --memory option used to start the container.</td>
<td>KPI</td>
</tr>
<tr>
<td>Used MB</td>
<td>Memory utilization by the container, in MB.</td>
<td>KPI</td>
</tr>
</tbody>
</table>

**Network**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default Monitoring Mode</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count (KB)</td>
<td>Network Volume, count (KB), incoming from an external network and outgoing to an external network.</td>
</tr>
<tr>
<td>Rate (KB/s)</td>
<td>Network Rate, rate (KB/s), incoming from an external network and outgoing to an external network</td>
</tr>
<tr>
<td>Incoming (KB)</td>
<td>Network traffic received by the container.</td>
</tr>
<tr>
<td>Outgoing (KB)</td>
<td>Network traffic sent by the container.</td>
</tr>
<tr>
<td>Incoming (KB/sec)</td>
<td>Rate of network traffic received by the container.</td>
</tr>
<tr>
<td>Outgoing (KB/sec)</td>
<td>Rate of network traffic sent by the container.</td>
</tr>
<tr>
<td>Incoming Packets</td>
<td>Number of packets received by the container.</td>
</tr>
<tr>
<td>Outgoing Packets</td>
<td>Number of packets sent by the container.</td>
</tr>
<tr>
<td>Incoming Packets Dropped</td>
<td>Number of incoming packets dropped before reaching the container. A high count of dropped packets might indicate high congestion or an issue on the network path.</td>
</tr>
<tr>
<td>Outgoing Packets Dropped</td>
<td>Number of outgoing packets dropped after leaving the container. A high count of dropped packets might indicate high congestion or an issue on the network path.</td>
</tr>
<tr>
<td>Incoming Errors</td>
<td>Number of packets received with errors.</td>
</tr>
<tr>
<td>Outgoing Errors</td>
<td>Number of packets sent with errors.</td>
</tr>
<tr>
<td>&lt;eth0&gt; Incoming Errors</td>
<td>Number of collision errors when &lt;eth0&gt; could not receive a frame due to a problem on the interface.</td>
</tr>
<tr>
<td>&lt;eth0&gt; Outgoing Errors</td>
<td>Number of collision errors when &lt;eth0&gt; could not send a frame due to a problem on the interface.</td>
</tr>
<tr>
<td>&lt;eth0&gt; Incoming KB</td>
<td>Average KB of traffic received on &lt;eth0&gt;</td>
</tr>
<tr>
<td>&lt;eth0&gt; Outgoing KB</td>
<td>Average KB of traffic sent on &lt;eth0&gt;</td>
</tr>
<tr>
<td>&lt;eth0&gt; Incoming KB/sec</td>
<td>Average rate of traffic received on &lt;eth0&gt;</td>
</tr>
<tr>
<td>&lt;eth0&gt; Outgoing KB/sec</td>
<td>Average rate of traffic sent on &lt;eth0&gt;</td>
</tr>
<tr>
<td>&lt;eth0&gt; Incoming Errors</td>
<td>Number of packets received on &lt;eth0&gt; with errors</td>
</tr>
<tr>
<td>&lt;eth0&gt; Outgoing Errors</td>
<td>Number of packets sent on &lt;eth0&gt; with errors</td>
</tr>
<tr>
<td>&lt;eth0&gt; Incoming Packets Dropped</td>
<td>Number of incoming packets dropped before reaching &lt;eth0&gt;. A high count of dropped packets might indicate high congestion or an issue on the network path.</td>
</tr>
<tr>
<td>&lt;eth0&gt; Outgoing Packets Dropped</td>
<td>Number of outgoing packets dropped after leaving &lt;eth0&gt;. A high count of dropped packets might indicate high congestion or an issue on the network path.</td>
</tr>
</tbody>
</table>