



APPDYNAMICS

AppDynamics Extensions and Integrations

AppDynamics Pro Documentation

Version 3.8.x

1. AppDynamics Extensions and Integrations	3
1.1 Add Metrics with a Monitoring Extension	5
1.1.1 Build a Monitoring Extension Using Java	6
1.1.2 Standalone Machine Agent HTTP Listener	19
1.1.3 Build a Monitoring Extension Using Scripts	20
1.2 Build an Alerting Extension	27
1.3 Use the AppDynamics REST API	35
1.3.1 Example Integrations Using the AppDynamics REST API	80
1.3.2 Use Curl to Access AppDynamics REST APIs	81
1.4 Integrate with AppDynamics for Databases	82
1.5 Configure Integrations	83
1.6 Integrate AppDynamics with Apica	83
1.7 Integrate AppDynamics with BMC End User Experience Management	84
1.8 Integrate AppDynamics with DB CAM	93
1.9 Integrate AppDynamics with Splunk	96
1.10 Configure Custom Metrics for the z-OS Machine Agent	100
1.11 Connector Development Guide	102

AppDynamics Extensions and Integrations

- [Add Metrics to AppDynamics Using Monitoring Extensions](#)
 - [Available Monitoring Extensions](#)
 - [Creating a Monitoring Extension](#)
- [Alerting Extensions](#)
 - [Available Alerting Extensions](#)
 - [Creating an Alerting Extension](#)
- [Performance Testing Extensions](#)
 - [Available Performance Testing Extensions](#)
 - [Create a Performance Testing Extension](#)
- [Cloud Auto-Scaling Extensions](#)
 - [Available Cloud Auto-Scaling Extensions](#)
 - [Creating a Cloud Auto-Scaling Extension](#)
- [Ecosystem Integrations](#)
- [Retrieve Data from AppDynamics Using the REST API](#)
- [Add AppDynamics Events](#)
- [Learn More](#)

AppDynamics provides many ways for you to extend AppDynamics Pro and integrate metrics with other systems.

Add Metrics to AppDynamics Using Monitoring Extensions

Using the standalone Machine Agent you can create extensions that add metrics to the set that AppDynamics already collects and reports to the Controller. These can include metrics that you collect from other monitoring systems. They can also include metrics that your system extracts from services that are not instrumented by AppDynamics, such as databases, LDAP servers, web servers, C programs, etc.

Available Monitoring Extensions

Many monitoring extensions are already available. Go to the [AppDynamics Exchange](#) and view extensions in the Monitoring category.

Creating a Monitoring Extension

To learn how to write your own monitoring extensions see [Add Metrics with a Monitoring Extension](#).

Alerting Extensions

Available Alerting Extensions

Many alerting extensions are already available. Go to the [AppDynamics Exchange](#) and view extensions in the Alerting category.

See also [Configure Integrations](#).

Creating an Alerting Extension

You can create a custom notification to integrate AppDynamics health rule violations and events with your alerting or ticketing system. See [Alert and Respond](#).

To learn how to write your own custom notification see [Build an Alerting Extension](#).

Performance Testing Extensions

Available Performance Testing Extensions

Many performance testing extensions are already available. Go to the [AppDynamics Exchange](#) and view extensions in the Performance Testing category.

- [Integrate AppDynamics with Apica](#)
- [Integrate with Soasta](#)
- [Integrate with HP LoadRunner](#)

Create a Performance Testing Extension

To learn how to write your own performance testing extension see [Use the AppDynamics REST API](#).

Cloud Auto-Scaling Extensions

AppDynamics works with other vendors to provide system integrations.

Available Cloud Auto-Scaling Extensions

Many cloud extensions are already available. Go to the [AppDynamics Exchange](#) and view extensions in the Cloud Connector category.

For instructions on registering compute clouds, see [Compute Clouds](#).

Creating a Cloud Auto-Scaling Extension

To learn how to write your own cloud auto-scaling extension see [Create Your Own Compute Cloud Connector](#).

Ecosystem Integrations

Additional integrations include:

- [Integrate with AppDynamics for Databases](#)
- [Integrate AppDynamics with BMC End User Experience Management](#)
- [Integrate AppDynamics with DB CAM](#)
- [Integrate AppDynamics with Splunk](#)

Retrieve Data from AppDynamics Using the REST API

You can programmatically retrieve data from AppDynamics and use it elsewhere, for example:

- Passing it to an alerting system, unified monitoring portal, phone app, etc.
- Pushing it to a corporate warehouse with other corporate data.
- Passing it to a tool that analyzes systems in your organization.

You can retrieve any metric available in the AppDynamics Metric Browser using the REST API, such as:

- Metadata that describes your applications, tiers, nodes, business transactions, etc.
- Real data such as metrics, events, transaction snapshots, health rule violations, etc.

To learn how to retrieve additional metrics and data, see [Use the AppDynamics REST API](#).

Add AppDynamics Events

You can create custom events using POST requests in the REST API. These can be events of type APPLICATION_DEPLOYMENT or of type CUSTOM.

You can create application deployment events, in addition to the ones that AppDynamics provides, to notify AppDynamics when you upgrade your application, push new code, etc. This lets you correlate these application deployment activities with other data inside AppDynamics. This is useful for regression analysis, root cause analysis, and performance studies. A useful practice is to include injection of your application deployment event into AppDynamics as part of the build process for deploying a new version of your application.

You can create custom events to be reported in the AppDynamics event viewer and in the events panels on the AppDynamics dashboards. Then you can create alerts triggered by these events as you do for AppDynamics standard events.

To learn how to add your own events see [Create Events](#).

Learn More

- [Build an AppDynamics Extension](#)
- [Request an Extension](#)

Add Metrics with a Monitoring Extension

- [About Custom Metrics](#)
- [Type of Monitoring Extensions](#)

About Custom Metrics

You can supplement the existing metrics in the AppDynamics Controller UI with your own custom metrics. 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, where you can display external metrics along with AppDynamics metrics on the same graph

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.

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.

Type of Monitoring Extensions

You can implement custom metrics using the following mechanisms.

- **Using a script:** You can write a shell script (LINUX) 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 may be too complicated to collect using a script. For example, you may 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.
- **Using Performance Counters in a .NET environment:** You can add additional performance counters to the standard hardware metrics reported by the embedded .NET Embedded Machine Agent by configuring the <perf-counters> element in the .NET Embedded Machine Agent's application.config file.
See [Enable Monitoring for Windows Performance Counters](#).

Build a Monitoring Extension Using Java

- [Machine Agent Required](#)
- [Before You Begin](#)
- [Process for Creating a Monitoring Extension in Java](#)
- [Your Monitoring Extension Class](#)
 - [Metric Path](#)
 - [Metric Name](#)
 - [Metric Processing](#)
 - [Aggregation](#)
 - [Time Roll Up](#)
 - [Cluster Roll Up](#)

- [Sample Monitoring Extension Class](#)
- [The monitor.xml File](#)
 - [Sample monitor.xml Files](#)
- [Using Your Monitoring Extension as a Task in a Workflow](#)

A Java monitoring extension enables the AppDynamics Machine Agent to collect custom metrics, which you define and provide, and to 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 the procedure for creating a monitoring extension in Java.

Machine Agent Required

A monitoring extension requires a standalone Machine Agent installed on the machine that hosts the application that you want to monitor.

If you do not know whether you have a Machine Agent installed:

1. In the upper right corner of AppDynamics console, click **Setup**.
2. Click **AppDynamics Agents**.

The list of agents appears with the Machine Agents below the app agents. You can get summary information about the machine and the application associated with each machine agent.

If you do not already have a Machine Agent installed, install one. See [Install the Standalone Machine Agent](#).

To the Machine Agent, a monitoring extension is a task that runs on a fixed schedule and collects metrics. The task can be either `AJavaTask`, which executes a task within the machine agent process, or `AForkedJavaTask`, which executes a task in its own separate process.

Before You Begin

Before creating your own extension from scratch, look at the extensions that have been created and shared among members of 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 exactly what you need or something close enough that you can download it and use it after making a few simple modifications.

Process for Creating a Monitoring Extension in Java

To create a monitoring extension in Java:

1. Create your extension class. See [Your Monitoring Extension Class](#).

2. Create a monitor.xml configuration file. See [The monitor.xml File](#).
3. Create a zip file containing these two files plus any dependent jar files.
4. Create a subdirectory (<your_extension_dir>) in your AppDynamics Machine Agent directory under Machine Agent/monitors.
5. Unzip the zip file into Machine Agent/monitors/<your_extension_dir>.
6. Restart the Machine Agent.

Your Monitoring Extension Class

Create a monitoring extension class by extending the AManagedMonitor class in the com.singularity.ee.agent.systemagent.api package.

You will also need the following helper classes in the same package:

- com.singularity.ee.agent.systemagent.api.MetricWriter;
- com.singularity.ee.agent.systemagent.api.TaskExecutionContext;
- com.singularity.ee.agent.systemagent.api.TaskOutput;
- com.singularity.ee.agent.systemagent.api.exception.TaskExecutionException

The Javadoc for these APIs is available at:

<https://rawgithub.com/Appdynamics/java-sdk/master/machine-agent-api/MachineAgentAPIJavado cs/index.html>

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.
- Describes the metrics using the MetricWriter class.
- Uploads the metrics to the Controller using the execute() method of the TaskOutput class.

Metric Path

All custom metrics processed by the Machine Agent appear in the Application Infrastructure Performance tree in the metric hierarchy.

Within the Application Infrastructure Performance tree you specify the metric path, which is the position of the metric in the metric hierarchy, using the "|" character. The first step in the metric path must be "Custom Metrics."

For example:

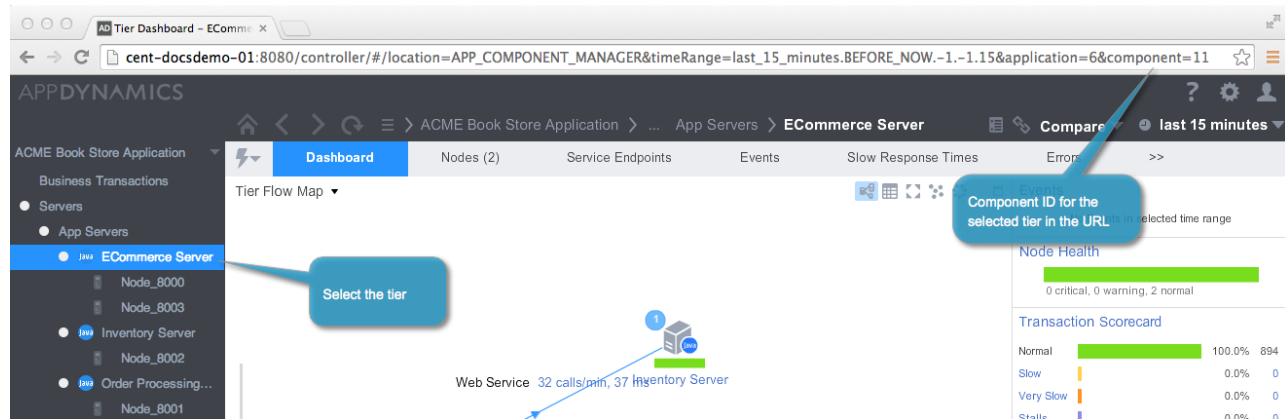
- Custom Metrics|WebServer|
- Custom Metrics|WebServer|XXX|, CustomMetrics|WebServer|YYY|

If the metrics apply to a specific tier, use the metric path for the tier, with "Component" followed by a colon ":" and the tier ID. The metric will appear under the specified tier in the metric path. For example:

- Server|Component:18|
- Server|Component:18|CustomMetric|Path|

To discover the component ID of a tier, select the tier from the Controller UI application tree. The

component ID appears in the URL of the browser.



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:18|JMX|Pool|First|pool usage,value="\$pool_value

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 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 Name

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 they will be 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

The Controller has various qualifiers for how it processes a metric with regard to aggregation, time rollup and tier rollup. You specify these options with the enumerated types provided by the MetricWriter class. These types are defined below.

Aggregation

The aggregator qualifier specifies how the values reported during a one-minute period are aggregated.

Aggregator Type	Description
METRIC_AGGREGATION_TYPE_AVERAGE	Average of all reported values in the minute. The default operation.

METRIC_AGGREGATION_TYPE_SUM	Sum of all reported values in the minute. This operation causes the metric to behave like a counter.
METRIC_AGGREGATION_TYPE_OBSERVATION	Last reported value in the minute. If no value is reported in that minute, the value from the last time it was reported is used.

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.

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

Cluster Roll Up

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

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

For example, if a tier has two nodes, Node A and Node B, and Node A has 3 errors per minute and Node B has 7 errors per minute, the INDIVIDUAL qualifier reports a value of 5 errors per minute and 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
- Requests: total number of requests

- Reading: number of reads
- Writing: number of writes
- Waiting: number of keep-alive connections

Here is the source for the extension class.

```
package com.appdynamics.monitors.nginx;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.StringReader;
import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.log4j.Logger;

import com.singularity.ee.agent.systemagent.api.AManagedMonitor;
import com.singularity.ee.agent.systemagent.api.MetricWriter;
import com.singularity.ee.agent.systemagent.api.TaskExecutionContext;
import com.singularity.ee.agent.systemagent.api.TaskOutput;
import com.singularity.ee.agent.systemagent.api.exception.TaskExecutionException;
import com.singularity.ee.util.httpclient.HttpClientWrapper;
import com.singularity.ee.util.httpclient.HttpExecutionRequest;
import com.singularity.ee.util.httpclient.HttpExecutionResponse;
import com.singularity.ee.util.httpclient.HttpOperation;
import com.singularity.ee.util.httpclient.IHttpClientWrapper;
import com.singularity.ee.util.log4j.Log4JLogger;

/**
 * NGinXStatusMonitor is a class that provides metrics on NGinX
server by using the
 * NGinX status stub.
 */
public class NGinXMonitor extends AManagedMonitor
{
    /**
     * The metric prefix indicates the metric's position in the
AppDynamics metric hierarchy.
     * It is prepended to the metric name when the metric is uploaded
to the Controller.
     * These metrics can be found in the AppDynamics metric hierarchy
under
     * Application Infrastructure Performance|{@literal <}Node{@literal
>}|Custom Metrics|WebServer|NGinX|Status
     */
    private final static String metricPrefix = "Custom
```

```
Metrics|WebServer|NGinx|Status|";

    /* Needed to connect to the Controller. Some environments may
    also require credentials. */
    protected volatile String host;
    protected volatile String port;
    protected volatile String location;

    /* Hash map to store metric values obtained from the source,
    in this example the NGinx server. */
    private Map<String,String> resultMap = new HashMap<String,String>();

    protected final Logger logger =
    Logger.getLogger(this.getClass().getName());

    /**
     ** Main execution method that uploads the metrics to the
    AppDynamics Controller.
     ** @see
    com.singularity.ee.agent.systemagent.api.ITask#execute(java.util.Map,
    com.singularity.ee.agent.systemagent.api.TaskExecutionContext)
     */
    public TaskOutput execute(Map<String, String> arg0,
    TaskExecutionContext arg1)
        throws TaskExecutionException
    {
        try
        {
            host = arg0.get("host");
            port = arg0.get("port");
            location = arg0.get("location");

            /* Gets the values for the metrics and
            populates the resultsMap. */
            populate();

            /* Outputs the metrics: metric name, value
            and processing qualifiers. */
            printMetric("Activity|up", 1,
            MetricWriter.METRIC_AGGREGATION_TYPE_OBSERVATION,
            MetricWriter.METRIC_TIME_ROLLUP_TYPE_SUM,
            MetricWriter.METRIC_CLUSTER_ROLLUP_TYPE_COLLECTIVE);

            printMetric("Activity|Active Connections",
            resultMap.get("ACTIVE_CONNECTIONS"),
            MetricWriter.METRIC_AGGREGATION_TYPE_OBSERVATION,
            MetricWriter.METRIC_TIME_ROLLUP_TYPE_CURRENT,
            MetricWriter.METRIC_CLUSTER_ROLLUP_TYPE_INDIVIDUAL
            );
        }
    }
}
```

```
        printMetric("Activity|Server|Accepts",
resultMap.get("SERVER_ACCEPTS"),
        MetricWriter.METRIC_AGGREGATION_TYPE_OBSERVATION,
        MetricWriter.METRIC_TIME_ROLLUP_TYPE_AVERAGE,
        MetricWriter.METRIC_CLUSTER_ROLLUP_TYPE_COLLECTIVE
    );

    printMetric("Activity|Server|Handled",
resultMap.get("SERVER_HANDLED"),
        MetricWriter.METRIC_AGGREGATION_TYPE_OBSERVATION,
        MetricWriter.METRIC_TIME_ROLLUP_TYPE_AVERAGE,
        MetricWriter.METRIC_CLUSTER_ROLLUP_TYPE_COLLECTIVE
    );

    printMetric("Activity|Server|Requests",
resultMap.get("SERVER_REQUESTS"),
        MetricWriter.METRIC_AGGREGATION_TYPE_OBSERVATION,
        MetricWriter.METRIC_TIME_ROLLUP_TYPE_AVERAGE,
        MetricWriter.METRIC_CLUSTER_ROLLUP_TYPE_COLLECTIVE
    );

    printMetric("Activity|Reading", resultMap.get("READING"),
        MetricWriter.METRIC_AGGREGATION_TYPE_OBSERVATION,
        MetricWriter.METRIC_TIME_ROLLUP_TYPE_AVERAGE,
        MetricWriter.METRIC_CLUSTER_ROLLUP_TYPE_COLLECTIVE
    );

    printMetric("Activity|Writing", resultMap.get("WRITING"),
        MetricWriter.METRIC_AGGREGATION_TYPE_OBSERVATION,
        MetricWriter.METRIC_TIME_ROLLUP_TYPE_AVERAGE,
        MetricWriter.METRIC_CLUSTER_ROLLUP_TYPE_COLLECTIVE
    );

    printMetric("Activity|Waiting", resultMap.get("WAITING"),
        MetricWriter.METRIC_AGGREGATION_TYPE_OBSERVATION,
        MetricWriter.METRIC_TIME_ROLLUP_TYPE_AVERAGE,
        MetricWriter.METRIC_CLUSTER_ROLLUP_TYPE_COLLECTIVE
    );
    /* Upload the metrics to the Controller. */
    return new TaskOutput("NGinX Metric Upload Complete");
}
catch (Exception e)
{
    return new TaskOutput("Error: " + e);
}
}

/**
 * Fetches Statistics from NGinX Server
 * @throws InstantiationException
```

```
* @throws IllegalAccessException
* @throws ClassNotFoundException
* @throws IOException
*/
protected void populate() throws InstantiationException,
    IllegalAccessException, ClassNotFoundException, IOException
{
    IHttpClientWrapper httpClient = HttpClientWrapper.getInstance();

    HttpExecutionRequest request = new
HttpExecutionRequest(getConnectionURL(), "", HttpOperation.GET);
    HttpExecutionResponse response =
httpClient.executeHttpOperation(request, new Log4JLogger(logger));

    Pattern numPattern = Pattern.compile("\\d+");
    Matcher numMatcher;

    BufferedReader reader = new BufferedReader(new
StringReader(response.getResponseBody()));
    String line, whiteSpaceRegex = "\\s";

    while ((line=reader.readLine()) != null)
    {
        if (line.matches("Active connections"))
        {
            numMatcher = numPattern.matcher(line);
            numMatcher.find();
            resultMap.put("ACTIVE_CONNECTIONS", numMatcher.group());
        }
        else if (line.matches("server"))
        {
            line = reader.readLine();

            String[] results = line.trim().split(whiteSpaceRegex);

            resultMap.put("SERVER_ACCEPTS", results[0]);
            resultMap.put("SERVER_HANDLED", results[1]);
            resultMap.put("SERVER_REQUESTS", results[2]);
        }
        else if (line.contains("Reading"))
        {
            String[] results = line.trim().split(whiteSpaceRegex);
            resultMap.put("READING", results[1]);
            resultMap.put("WRITING", results[3]);
            resultMap.put("WAITING", results[5]);
        }
    }
}

/**
```

```
* Returns the metric to the AppDynamics Controller.
* @param metricName Name of the Metric
* @param metricValue Value of the Metric
* @param aggregation Average OR Observation OR Sum
* @param timeRollup Average OR Current OR Sum
* @param cluster Collective OR Individual
*
* Overrides the Metric Writer class printMetric() by building the
string that provides all the fields needed to
* process the metric.
*/
public void printMetric(String metricName, Object metricValue,
String aggregation, String timeRollup, String cluster)
{
    MetricWriter metricWriter = getMetricWriter(getMetricPrefix() +
metricName,
    aggregation,
    timeRollup,
    cluster
    );

    metricWriter.printMetric(String.valueOf(metricValue));
}

protected String getConnectionURL()
{
    return "http://" + host + ":" + port + "/" + location;
}

protected String getMetricPrefix()
{
    return metricPrefix;
}
```

```
}  
}
```

The monitor.xml File

Create a monitor.xml file with a <monitor> element to configure how the machine agent will execute 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.
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

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


```
<monitor>
  <name>NGinxMonitor</name>
  <type>managed</type>
  <description>NGinx server monitor</description>
  <monitor-configuration></monitor-configuration>
  <monitor-run-task>
    <execution-style>periodic</execution-style>

    <execution-frequency-in-seconds>60</execution-frequency-in-seconds>
    <name>NGinx Monitor Run Task</name>
    <display-name>NGinx Monitor Task</display-name>
    <description>NGinx Monitor Task</description>
    <type>java</type>

    <execution-timeout-in-secs>60</execution-timeout-in-secs>
    <task-arguments>
      <argument name="host" is-required="true"
default-value="localhost" />
      <argument name="port" is-required="true"
default-value="80" />
      <argument name="location" is-required="true"
default-value="nginx_status" />
    </task-arguments>
    <java-task>
      <classpath>NginxMonitor.jar</classpath>

    <impl-class>com.appdynamics.nginx.NGinxMonitor</impl-class>
    </java-task>
  </monitor-run-task>
</monitor>
```

The next 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.

```

<monitor>
  <name>MysqlMonitor</name>
  <enabled>true</enabled>
  <type>managed</type>
  <description>Monitors Mysql database system </description>
  <monitor-configuration></monitor-configuration>
  <monitor-run-task>
    <execution-style>periodic</execution-style>

    <execution-frequency-in-seconds>60</execution-frequency-in-seconds>
    <name>Mysql Monitor Run Task</name>
    <display-name>Mysql Monitor Task</display-name>
    <description>Mysql Monitor Task</description>
    <type>java</type>
    <execution-timeout-in-secs>60</execution-timeout-in-secs>
    <task-arguments>
      <argument name="host" is-required="true"
default-value="localhost" />
      <argument name="port" is-required="true"
default-value="3306" />
      <argument name="user" is-required="true"
default-value="root" />
      <argument name="password" is-required="true"
default-value="welcome" />

      <!--
      The tier under which the metrics should appear in the
metric browser.
      If this argument is left out then the metrics will be
registered in every tier.
      -->
      <argument name="tier" is-required="false"
default-value="1stTier" />
    </task-arguments>
  </java-task>

  <classpath>mysql.jar;mysql-connector-java-5.1.17-bin.jar</classpath>

  <impl-class>com.singularity.ee.agent.systemagent.monitors.database.m
ysql.MysqlMonitor</impl-class>
  </java-task>
</monitor-run-task>
</monitor>

```

Using Your Monitoring Extension as a Task in a Workflow

Your monitoring extension can be invoked as a task in a workflow if you upload the zip file to the task library. Use the instructions in [To package the XML files as a Zip archive](#) to upload the Java

monitor to the Task Library.

Standalone Machine Agent HTTP Listener

- To activate the HTTP listener
- To send metrics
- To send events
- To upload metrics
- To upload events
- To shut down the standalone machine agent

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.

To activate the HTTP listener

- Restart the Standalone Machine Agent using metric.http.listener:

```
java -Dmetric.http.listener=true -Dmetric.http.listener.port=<port_number>
-jar machineagent.jar
```

If you do not specify the optional metric.http.listener.port, it defaults to 8293.

To send metrics

```
GET | POST /machineagent/metrics
```

To send events

```
GET /machineagent/event
```

To 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. For example:

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

Valid values for type are:

- average
- sum
- current

To upload events

Send events using HTTP get requests to:

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

Event_type is one of the following:

- error
- info
- summary
- warning

To shut down the standalone machine agent

```
GET /machineagent/shutdown
```

Build a Monitoring Extension Using Scripts

- [Metric Qualifiers](#)
 - [Aggregation Qualifier](#)
 - [Time Roll Up Qualifier](#)
 - [Cluster Roll Up Qualifier](#)
- [Adding a Monitoring Extension](#)
 - [Step 1. Create a directory under the Machine Agent monitors directory](#)
 - [Step 2. Create the script file](#)
 - [Step 3. Copy the script file to the directory created in Step 1](#)
 - [Step 4. Create the monitor.xml file](#)
 - [Step 5. Copy the monitor.xml file to the directory created in Step 1](#)
 - [Step 6. Restart the standalone Machine Agent](#)
 - [Step 7. Verify execution of the monitoring extension script](#)
- [Example: Create a monitoring extension for open files](#)
- [Learn More](#)

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

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

Metric Qualifiers

The metric qualifiers determine how the Controller processes the data for your custom metric. There are three types:

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

Aggregation Qualifier

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

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

Time Roll Up Qualifier

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

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

Cluster Roll Up Qualifier

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

Roll up Strategy	Description
INDIVIDUAL	Aggregates the metric value by averaging the metric values across each node in the tier.

COLLECTIVE

Aggregates the metric value by adding up the metric values for all the nodes in the tier.

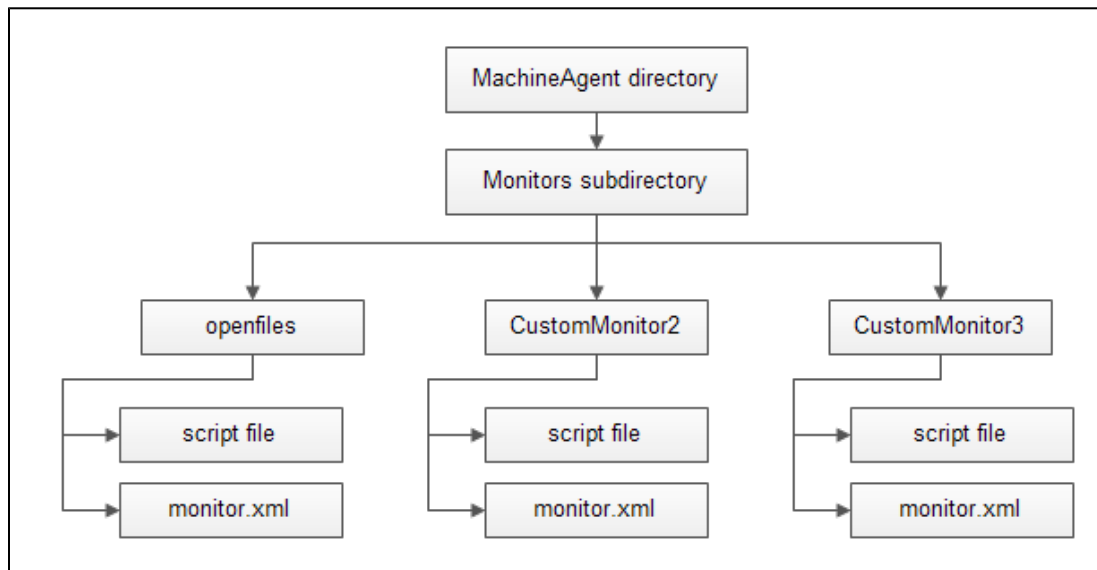
For example, if a tier has two nodes, Node A and Node B, and Node A has 3 errors per minute and Node B has 7 errors per minute, the INDIVIDUAL qualifier reports a value of 5 errors per minute and and COLLECTIVE qualifier reports 10 errors per minute. INDIVIDUAL is appropriate for metrics such as % CPU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

Adding a Monitoring Extension

Step 1. Create a directory under the Machine Agent monitors directory

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

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



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.

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

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

Use the following format:

	Format
Standard Form	Hardware Resources Instrument Name=Instrument Value
Fully Qualified Form	Hardware Resources <metric name>,value=<long value>

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

- Infrastructure (for the default hardware metrics, see [Monitor Hardware](#))
- JVM (for the default metrics, see [Monitor JVMs](#))
- Custom Metrics

Custom metrics must have the path prefixes:

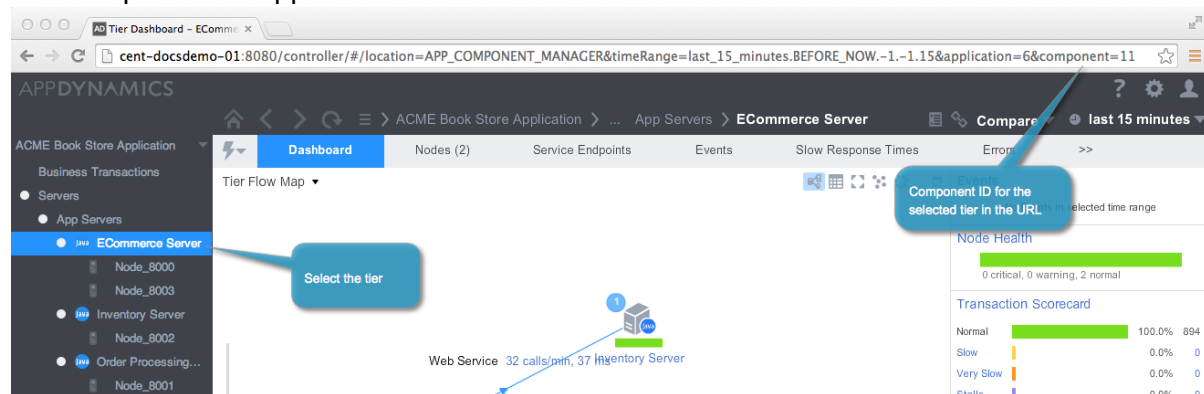
- Custom Metrics
- Server|Component:<id>

Metrics with the Custom Metrics prefix are common across all tiers in your application.

Metrics with the Server|Component:<id> prefix appear only under the specified tier.

To discover the component ID of a tier, select the tier from the Controller UI application tree.

The component ID appears in the URL of the browser.



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.

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

```
name=Custom Metrics|Hardware Resources|Disks|Total Disk Usage %,
value=23
name=Custom Metrics|Hardware Resources|Disks|Disk 1|Current Disk
Usage %, value=56
```

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

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

Step 4. Create the monitor.xml file

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

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

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

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

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

Execution Style	Description	Example
Continuous	Choose "Continuous", if you want data collection averaged over time. (For example: Averaging CPU over minute). This ensures that the script keeps running until the machine agent process is terminated.(!) For the monitor to be declared as 'continuous', the script should also run in an infinite loop.	while [1]; do ... the actual script goes here ... sleep 60 done
Periodic	Choose periodic, if you want data to be reported from system performance counters periodically.	The periodic task runs every minute by default. Use following parameter in the XML file, if you want the periodic task to run with any other frequency: <monitor-run-task> element. <execution-frequency-in-seconds>120</execution-frequency-in-seconds> For all the other frequency settings, the data is aggregated.

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

```
<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 directory created in Step 1

Step 6. Restart the standalone Machine Agent

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

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

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.

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

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

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

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

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

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

Learn More

- [Administer Machine Agents](#)
- [Notification Actions](#)
- [Machine Agent Install and Admin FAQ](#)
- [Supported Environments and Versions](#)

Build an Alerting Extension

- [Custom Notifications and Custom Actions](#)
- [Creating a Custom Action](#)
 - [To create a custom action script](#)
- [Contents of the custom.xml File](#)
- [Information Passed to the Custom Action Script from AppDynamics](#)
 - [Parameters passed by a health rule violation](#)
 - [Parameters passed for an event that is not a health rule violation event](#)
- [Sample Alerting Extension](#)
- [Learn More](#)

If you are using an on-premise controller, you can set up a custom action on the controller instance to integrate notification of AppDynamics health rule violations and events with an alerting or ticketing system. Use a push approach by creating custom notifications that pass the information to your alerting system.

Custom Notifications and Custom Actions

A custom notification lets you integrate alerts about AppDynamics health rule violations and events into your own alerting system. This integration extension requires:

- An on-premise controller
- A custom.xml file that provides information about the custom notification
- An executable script that accepts parameters from AppDynamics about the events and health rule violations that trigger an alert
- Configuring AppDynamics events or policies to trigger the custom notification via a custom action

Creating a Custom Action

To create a custom action script

1. At the top level of the Controller installation directory, create a directory named "custom" with a sub-directory named "actions".

```
<controller_install_dir>/custom/actions
```

2. In the <controller_install_dir>/custom/actions directory, create a subdirectory for each custom action script that you will install. For example,

```
<controller_install_dir>/custom/actions/jira
```

for an action that interfaces with a JIRA system.

3. For each custom action that you want to implement, create an executable script (.bat extension for Windows, .sh extension for Linux) that can accept and process the parameters passed to it by AppDynamics. See [Information Passed to the Custom Action Script from AppDynamics](#) for details on the parameters.

Create this script in the appropriate subdirectory that you created in step 2.

Set correct executable permissions for the shell scripts in a Linux environment. For example:
chmod 770 script1.sh.

Ensure that the script file has the correct character encoding. This is especially important when creating a Unix shell script on a Windows machine.

4. In the <controller_install_dir>/custom/actions directory, create a custom.xml file that describes the location and name of your custom action script(s).

See [Contents of the custom.xml File](#).

5. After you have installed the script(s) and the custom.xml file, restart the Controller.

6. Verify the script manually.

To verify the script:

- a. Open a command-line console on the Controller host machine.
- b. Execute the script file from the command line console.

7. Create the custom action in the AppDynamics UI. See [Custom Actions](#).

Contents of the custom.xml File

The custom.xml file has an <actions> element for every custom action on the controller.
The <type> element contains the subdirectory that contains the script file.
The <executable> element contains the name of the script.

Sample custom.xml file

```
<custom-actions>
  <action>
    <type>jira</type>
    <executable>script1.bat</executable>
  </action>
  <action>
    <type>bugzilla</type>
    <executable>script2.sh</executable>
  </action>
</custom-actions>
```

Information Passed to the Custom Action Script from AppDynamics

The custom action script must handle the parameters passed to it from a health rule violation event or any other type of event.

Parameters passed by a health rule violation

For a health rule violation, the custom action script is invoked with the following string parameters:

Health Rule Violation Parameter	Repeated Parameter	Notes
APP_NAME	none	name of the business application
APP_ID	none	application ID number
PVN_ALERT_TIME	none	alert time, such as: Thu Dec 22 15:03:56 PST 2011
PRIORITY	none	priority number
SEVERITY	none	allowed values: INFO, WARN, or ERROR (In the AppDynamics UI they are called "Info", "Warning", and "Critical")
TAG	none	or the empty string if no tag was specified by the user
HEALTH_RULE_NAME	none	name of the health rule that was violated

HEALTH_RULE_ID	none	health rule ID number
PVN_TIME_PERIOD_IN_MINUTES	none	health rule violation time period in minutes
AFFECTED_ENTITY_TYPE	none	allowed types: APPLICATION, APPLICATION_COMPONENT (aka Tier), APPLICATION_COMPONENT_NODE, BUSINESS_TRANSACTION, APPLICATION_DIAGNOSTIC_DATA (aka Error)
AFFECTED_ENTITY_NAME	none	the affected entity name
AFFECTED_ENTITY_ID	none	the affected entity id
NUMBER_OF_EVALUATION_ENTITIES	none	number of entities (BT, App, Tiers, Nodes, Errors, JMX counters, etc..) violating health rule conditions
EVALUATION_ENTITY_TYPE	Yes, one for each evaluation entity	allowed types: APPLICATION, APPLICATION_COMPONENT (aka Tier), APPLICATION_COMPONENT_NODE, BUSINESS_TRANSACTION, APPLICATION_DIAGNOSTIC_DATA (aka Error), JMX
EVALUATION_ENTITY_NAME	Yes, one for each evaluation entity	the evaluation entity name (for JMX it is the counter name)
EVALUATION_ENTITY_ID	Yes, one for each evaluation entity	the evaluation entity id or <NULL> for JMX
NUMBER_OF_TRIGGERED_CONDITIONS_PER_EVALUATION_ENTITY	Yes, one for each evaluation entity	number of times to loop through the triggered condition parameters for each evaluation entity
triggered condition	Yes, listed below	if more than one condition is triggered, the parameters repeat for each triggered condition, where "x" increments the number representing the condition

	SCOPE_TYPE_x	the scope of the parameter, whether the scope is the application, tier, or node: APPLICATION, APPLICATION_COMPONENT, APPLICATION_COMPONENT_NODE
	SCOPE_NAME_x	the name of the scope, such as: ACME Book Store Application
	SCOPE_ID_x	the scope id
	CONDITION_NAME_x	the health rule condition name
	CONDITION_ID_x	the health rule condition id
	OPERATOR_x	allowed operators: LESS_THAN, LESS_THAN_EQUALS, GREATER_THAN, GREATER_THAN_EQUALS, EQUALS, NOT_EQUALS.
	CONDITION_UNIT_TYPE_x	the condition for the threshold parameter: ABSOLUTE, BASELINE_STANDARD_DEVIATION, BASELINE_PERCENTAGE, BASELINE_PERCENTILE
	USE_DEFAULT_BASELINE_x	a Boolean parameter (true or false) applicable only when the condition unit type is one of the BASELINE_ types
	BASELINE_NAME_x	applicable only when the condition unit type is one of the BASELINE_ types and the use default baseline parameter is "false"
	BASELINE_ID_x	applicable only when the condition unit type is one of the BASELINE_ types and the use default baseline parameter is "false"
	THRESHOLD_VALUE_x	health rule threshold setting

	OBSERVED_VALUE_x	value that violated the health rule threshold
SUMMARY_MESSAGE	none	summary of the notification, such as: "Health rules have been violated."
INCIDENT_ID	none	the incident identifier number for this health rule violation
DEEP_LINK_URL	none	controller deep link URL, such as: http://<controller-host-url>/#location=APP_INCIDENT_DETAIL&incident= Append the incident ID to the URL to provide a link to the Controller UI for this policy violation
EVENT_TYPE	none	POLICY_OPEN_WARNING,POLICY_OPEN_CRITICAL,POLICY_CLOSE,POLICY_OPEN_UPGRADED',POLICY_OPEN_DOWNGRADED

Parameters passed for an event that is not a health rule violation event

For an event that is not a health rule violation event, the custom action script is invoked with the following string parameters:

Event Notification Parameter	Repeated Parameter	Notes
APP_NAME	none	name of the business application
APP_ID	none	application ID number
EN_TIME	none	event notification time, for example: Wed Jan 04 09:36:55 PST 2012
PRIORITY	none	priority number
SEVERITY	none	Allowed values: INFO, WARN, or ERROR (In the AppDynamics UI they are called "Info", "Warning", and "Critical")

TAG	none	or <NULL> if it was not specified by the user
EN_NAME	none	name of the event notification
EN_ID	none	event notification ID number
EN_INTERVAL_IN_MINUTES	none	event notification interval in minutes
NUMBER_OF_EVENT_TYPES	none	determines how many times to loop through the event type map parameters
event type	yes, listed below	if there is more than one event type, the parameters repeat for each event type, where "x" increments the number representing the event type
	EVENT_TYPE_x	type of event, such as: ERROR, APPLICATION_ERROR, APPLICATION_INFO, STALL, BT_SLA_VIOLATION, DEADLOCK, MEMORY_LEAK, MEMORY_LEAK_DIAGNOSTICS, LOW_HEAP_MEMORY, ALERT, CUSTOM, APP_SERVER_RESTART, BT_SLOW, SYSTEM_LOG, INFO_INSTRUMENTATION_VISIBILITY, AGENT_EVENT, INFO_BT_SNAPSHOT, AGENT_STATUS, SERIES_SLOW, SERIES_ERROR, ACTIVITY_TRACE, OBJECT_CONTENT_SUMMARY, DIAGNOSTIC_SESSION, HIGH_END_TO_END_LATENCY, APPLICATION_CONFIG_CHANGE, APPLICATION_DEPLOYMENT, AGENT_DIAGNOSTICS, MEMORY, LICENSE
	EVENT_TYPE_NUM_x	number of events of this type

NUMBER_OF_EVENT_SUMMARIES	none	number of event summaries in the notification; determines how many times to loop through the event summary parameters
event summary	yes, listed below	if there is more than one event summary, the following parameters repeat for each event summary, where "x" increments the number representing the event summary
	EVENT_SUMMARY_ID_x	event summary ID number
	EVENT_SUMMARY_TIME_x	event summary time, for example: Wed Jan 04 09:34:13 PST 2012
	EVENT_SUMMARY_TYPE_x	type of event, such as: APPLICATION_CONFIG_CHANGE, APP_SERVER_RESTART, DIAGNOSTIC_SESSION, STALL
	EVENT_SUMMARY_SEVERITY_x	event severity, such as: INFO, WARN, or ERROR (In the AppDynamics UI they are called "Info", "Warning", and "Critical")
	EVENT_SUMMARY_STRING_x	event summary string, such as: Application Server environment variables changed
DEEP_LINK_URL	none	http://<controller-host-url>/#location=APP_EVENT_VIEWER_MODAL&eventSummary=Append each event summary ID to the URL to provide a link to the Controller UI for this event

The current custom action implementation will most likely pass more than nine parameters to the executing script.

In Windows and Linux environments, more than nine command line parameters can be processed only if the SHIFT command is used.

For more information, see [Command Line parameters](#).

Sample Alerting Extension

You can [download an alerting extension](#) that interfaces with the PagerDuty alerting and incident tracing system.

Learn More

- [Actions](#)
- [Custom Actions](#)
- [Policies](#)
- [AppDynamics Extensions and Integrations](#)
- [Configure Policies](#)

Use the AppDynamics REST API

- [Introduction](#)
- [Authentication](#)
 - [Invalid Characters for Usernames](#)
- [Retrieve all business applications](#)
 - [Example - Retrieve list of all business applications.](#)
- [Retrieve all Business Transactions in a particular business application](#)
 - [Example - Retrieve list of all business transactions for the ACME Book Store.](#)
- [Retrieve all tiers in a business application](#)
 - [Example - Retrieve the list of all tiers for the ACME Book Store.](#)
- [Retrieve machine, agent, and IP information about all nodes in a business application](#)
 - [Example - Retrieve machine, agent and IP information about the nodes in application 3](#)
- [Retrieve machine, agent, and IP information about a node by node name](#)
 - [Example - Retrieve information about Node_8001 in application 3.](#)
- [Retrieve machine, agent, and IP information about all the nodes in a tier](#)
 - [Example - Retrieve information about the nodes in the E-Commerce tier in the ACME Online Book Store.](#)
- [Retrieve information about a tier, including the tier ID and number of nodes, in a tier by tier name](#)
 - [Example - Retrieve information about the ECommerce tier in the ACME Online Book Store.](#)
- [Retrieve metrics](#)
 - [To Copy the REST URL for a Metric](#)
 - [To Copy the Metric-Path Parameter](#)
 - [Structure of Returned Metrics](#)
 - [Values of Returned Metrics](#)
 - [Retrieve metric hierarchy](#)
 - [Example - Retrieve the metric hierarchy for the ACME Book Store.](#)
 - [Use Wild cards in Metric-Path Parameter](#)
 - [Example - Retrieve the average response time for all the tiers in the application using the wild card character for the tier name.](#)
 - [Example - Retrieve the CPU %Busy metric for all the nodes in the ECommerce tier using the wild card character for the node name.](#)
 - [Example - Retrieve the CPU %Busy metric for all the nodes in all the tiers using wild card characters for the tier and node names.](#)

- Example - Retrieve the Calls per Minute metric for all the business transactions on the ECommerce tier using the wild card character for the business transaction name
- Retrieve metrics for a time range
 - Example - Retrieve the average response time for the past 15 minutes on the ECommerce server.
 - Example - Retrieve the multiple metrics, for the past 15 minutes, for the ViewCart.sendItems transaction on the ECommerce server.
 - Example - Retrieve the average cpu used by the All Other Traffic business transaction during the past 15 minutes on the ECommerce server.
 - Example - Retrieve snapshots for the 15 minutes after Mon, 13 Aug 2012 08:20:41 for the ACME Book Store Application.
 - Example - Retrieve snapshots for the 15 minutes before Mon, 13 Aug 2012 08:20:41 for the ACME Book Store Application.
 - Example - Retrieve snapshots for the time range between Mon, 13 Aug 2012 08:20:41 and Mon, 13 Aug 2012 08:22:21 GMT for the ACME Book Store Application.
- Retrieve all health rule violations in a particular business application
- Retrieve all policy violations in a particular business application
 - Example - Retrieve the list of all policy violations in the ACME Book Store for the past hour.
- Retrieve event data
 - Example - Retrieve the list of events of type "APPLICATION_ERROR" or "DIAGNOSTIC_SESSION" of any severity that occurred in the specified time range.
- Create Events
 - Application Deployment Event Integration
 - Create a Custom Event
- Retrieve transaction snapshots for a Business Transaction for a time range
 - Example - Retrieve list of transaction snapshots for the ACME Book Store.
 - Example - Retrieve list of transaction snapshots including the snapshot fields that are associated with an Http Parameter data collector.
- Create and modify AppDynamics users
- Include or exclude a business transaction from monitoring
 - Example - Exclude business transaction 166 from monitoring.
- Retrieve all controller global configuration values
 - Example - Retrieve list of all global configuration values
- Retrieve a single controller global configuration value
 - Example - Retrieve the global metrics buffer size.
- Configure Global Controller Settings
- Mark Nodes as Historical
 - Example - Mark nodes 44 and 45 as historical.
- Retrieve all policy violations in a particular business application
 - Example - Retrieve the list of all open policy violations in the ACME Book Store for the past 1000 minutes.

Introduction

You use the REST API to retrieve information from AppDynamics programmatically.

The AppDynamics REST API is implemented using Representational State Transfer (REST)

Services. The data can be returned in either the JavaScript Object Notation (JSON) or the eXtensible Markup Language (XML) format. The default output format is XML.

The URIs in the Monitor APIs are a set of REST services that open access to Monitor data collected by AppDynamics. Each URI can be found by accessing:

```
http://<Controller_Host>:<Controller_Port>/controller/rest/<REST_URI>
```

You can find more general information at [REST on Wikipedia](#) and in particular [RESTful web APIs](#).

You may find the [REST Python Client Extension](#) to be a useful tool.

For information about importing and exporting transaction detection configurations using REST, see [Import and Export Transaction Detection Configuration \(Java\)](#).

Authentication

To invoke the REST APIs, provide basic HTTP authentication credentials as well as your account information. These are:

- Account: the AppDynamics tenant account name
- Username: a user in that account
- Password: the password for that account

If you have installed an on-premise Controller on a single-tenant platform or if you are using the AppDynamics SaaS Controller, your default account is "customer1". In this case the username to log in with would be:

```
<your_username>@customer1
```

If you are using a multi-tenant Controller, you need to log in with a username in your multi-tenant account:

```
<your_username>@<your_accountname>
```

Invalid Characters for Usernames

Usernames that contain the following characters are not authenticated for REST API calls:

```
\ / " [ ] : | < > + = ; , ? * @, ' tab space
```

If you have already created usernames that contain any of the disallowed characters, such as "user:customer66", create a new username without the disallowed character for the purpose of accessing the REST APIs.

Retrieve all business applications

URI: /controller/rest/applications

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve list of all business applications.

URI	Sample object output
/controller/rest/applications	XML
/controller/rest/applications?output=JSON	JSON

Retrieve all Business Transactions in a particular business application

URI: /applications/<application-name | application-id>/business-transactions

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes
exclude	Query	If false, the query retrieves only the business transactions that are included for monitoring. If true, the query retrieves only the excluded business transactions. Excluded business transactions are those that have been configured to be excluded from monitoring either from the UI or through the REST interface. The default is false.	No

Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No
--------	-------	--	----

Example - Retrieve list of all business transactions for the ACME Book Store.

URI	Sample object output
/controller/rest/applications/ACME Book Store Application/business-transactions	XML
/controller/rest/applications/ACME Book Store Application/business-transactions?output=JSON	JSON

Retrieve all tiers in a business application

URI: /controller/rest/applications/<application-name|application-id>/tiers

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve the list of all tiers for the ACME Book Store.

URI	Sample object output
/controller/rest/applications/ACME Book Store Application/tiers	XML
/controller/rest/applications/ACME Book Store Application/tiers?output=JSON	JSON

Retrieve machine, agent, and IP information about all nodes in a business application

URI: /controller/rest/applications/<application-name|application-id>/nodes

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve machine, agent and IP information about the nodes in application 3

URI	Sample object output
/controller/rest/applications/3/nodes	XML

Retrieve machine, agent, and IP information about a node by node name

URI: /controller/rest/applications/<application-name | application-id>/nodes/<node-name>

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes
<node-name>	URI	Provide the node name	Yes

Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No
--------	-------	---	----

Example - Retrieve information about Node_8001 in application 3.

URI	Sample object output
/controller/rest/applications/3/nodes/Node_8001	XML

Retrieve machine, agent, and IP information about all the nodes in a tier

URI: /controller/rest/applications/<application-name | application-id>/tiers/<tier-name>/nodes

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes
<tier-name>	URI	Provide the tier name.	Yes
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve information about the nodes in the E-Commerce tier in the ACME Online Book Store.

URI	Sample object output
/controller/rest/applications/ACME Online Book Store/tiers/E-Commerce/nodes/	XML

Retrieve information about a tier, including the tier ID and number of nodes, in a tier by tier name

URI: /controller/rest/applications/<application-name | application-id>/tiers/<tier-name>

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes
<tier-name>	URI	Provide the tier name.	Yes
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve information about the ECommerce tier in the ACME Online Book Store.

URI	Sample object output
/controller/rest/applications/ACME Online Book Store/tiers/ECommerce	XML
/controller/rest/applications/ACME Online Book Store/tiers/ECommerce?output=JSON	JSON

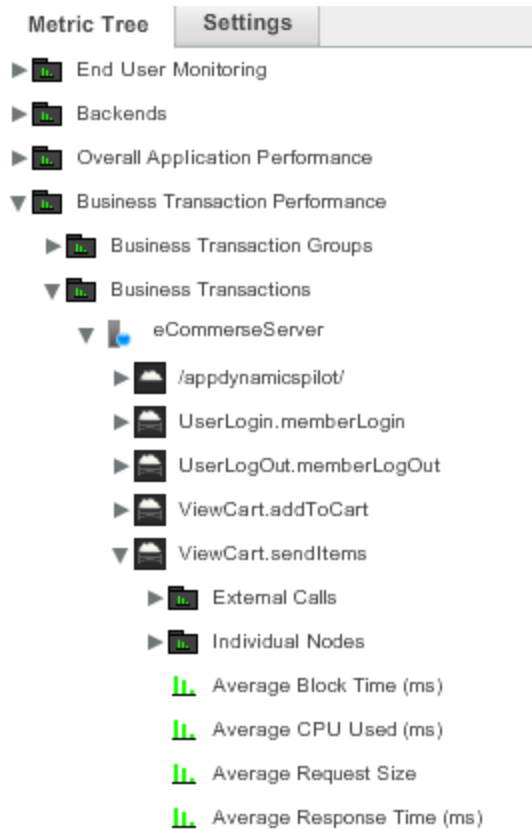
Retrieve metrics

AppDynamics groups the metrics that it collects into following categories:

- Backends
- End User Monitoring
- Overall Application Performance
- Business Transaction Performance
- Application Infrastructure Performance
- Errors
- Information Points

To see the structure of the metric hierarchy of a business application in the AppDynamics UI, expand the nodes in the Metric Browser:

1. In the left navigation pane, select the application for which you are retrieving metrics.
2. Click **Analyze -> Metric Browser**.
3. In left panel expand the nodes in the metric tree.



You can copy the URI for fetching any metric directly from any node in the Metric Browser. This is the easiest way to construct a query to retrieve a particular metric. You can also copy just the metric path portion of the query.

The child elements in the metric path expression are separated by the pipe character (|). The pipe character must not appear at the beginning or end of the metric path expression.

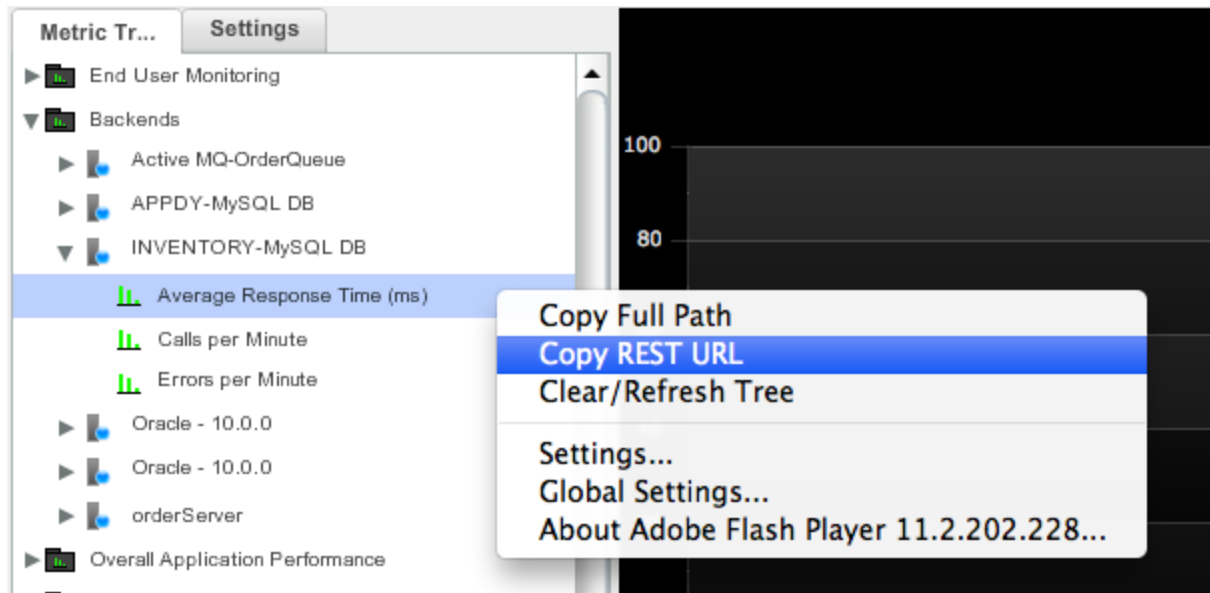
The easiest way to construct a query to retrieve a metric is to select the metric in Metric Browser, copy the REST URL, and paste it into your browser. An alternative is select the metric and copy the metric path, and construct the query by prefacing the metric path with "metric-data?metric-path=".

To Copy the REST URL for a Metric

To copy the REST URL for any metric captured by AppDynamics:

1. Open the Metric Browser as described above.
2. Select the item for which you want to retrieve metrics in the metric tree. You can retrieve the URL at any level of the tree.
3. Right-click the item and select **Copy REST URL** from the drop-down menu.
4. Paste the URL into your Web browser to run the query.

The following example copies the URL for the Average Response Time in the Inventory-MSQL DB for the Acme Online Book Store application for the time period that was selected in the metric browser.



The copied query is

```
http://ec2-23-20-107-243.compute-1.amazonaws.com:8090/controller/rest/applications/AcmeOnlineBookStore/metric-data?metric-path=Backends%7CINVENTORY-MySQL%20DB%7CAverage%20Response%20Time%20(ms)&time-range-type=BEFORE_NOW&duration-in-mins=15
```

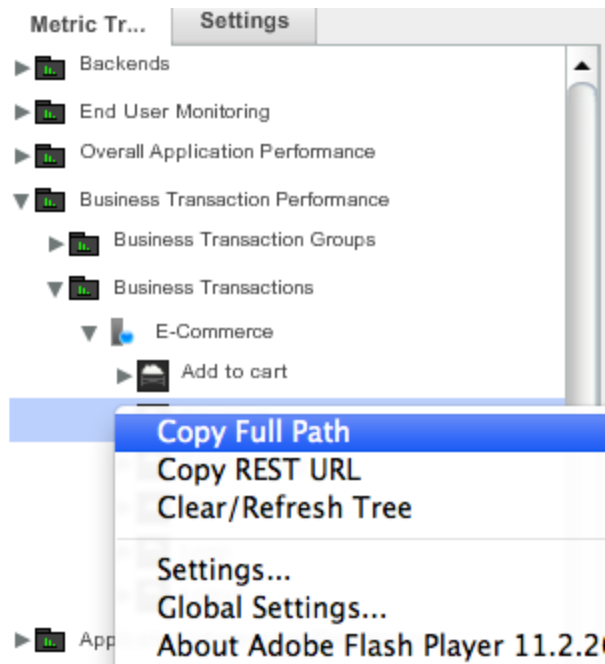
The time range in the request is based on the setting of the time range in the Metric Browser when you copied the URL. You can edit a copied query to change the time range and duration. See [Retrieve metrics for a time range](#) for more information.

To Copy the Metric-Path Parameter

You also copy only the metric path for a specific metric.

To copy a metric path:

1. Open the Metric Browser as described above.
2. Select the item for which you want to retrieve metrics in the metric tree. You can retrieve the metric path at any level of the tree.
3. Right-click the item and select **Copy Full Path** from the drop-down menu.
4. Use the metric path to construct your query.



The copied data is

```
Business Transaction Performance|Business Transactions|E-Commerce|Add
to cart
```

Structure of Returned Metrics

The data is returned in a tree structure. If a child element is a container item, its <type> tag is set to "folder". Otherwise the <type> tag for the child element is set to "leaf".

A folder <type> tag indicates that the metric has child elements. The API retrieves the first generation of child elements. You can expand only the children of the folder type.

Values of Returned Metrics

Results are returned in a metricValues structure that contains these fields:

- current
- value
- min
- max
- startTimestampInMillis

The "current" value is the value for the current minute.

The "value" value is one of the following for all metric values reported across the configured evaluation time length:

- arithmetic average, if the metric time rollup type is average
- sum, if the metric time rollup type is sum
- latest, if the metric time rollup type is current

The "min" and "max" values are the minimum and maximum values reported across the configured evaluation time length. These are not used for all metric types.

The `startTimeInMillis` is the start time of the time range to which the result metric data applies, in UNIX epoch time.

By default, the values of the returned metrics are rolled up into a single data point (`rollup=true`). To get separate results for all the values within the specified time range, set the `rollup` parameter to `false` in the query.

The returned `metricValues` structure typically looks like this:

XML output with `rollup=true`

```
<metricValues>
  <metric-value>
    <startTimeInMillis>1388524620000</startTimeInMillis>
    <value>244</value>
    <min>3</min>
    <max>10008</max>
    <current>10007</current>
  </metric-value>
</metricValues>
```

XML output with `rollup=false`

```
<metricValues>
  <metric-value>
    <startTimeInMillis>1388524620000</startTimeInMillis>
    <value>374</value>
    <min>13</min>
    <max>10007</max>
    <current>14</current>
  </metric-value>
  <metric-value>
    <startTimeInMillis>1388524680000</startTimeInMillis>
    <value>14</value>
    <min>13</min>
    <max>18</max>
    <current>13</current>
  </metric-value>
  <metric-value>
    <startTimeInMillis>1388524740000</startTimeInMillis>
    <value>14</value>
    <min>13</min>
    <max>18</max>
    <current>13</current>
  </metric-value>
  . . .
  . . .
```

JSON output with `rollup=true`

```
"metricValues": [  {
  "current": 14,
  "max": 10009,
  "min": 3,
  "startTimeInMillis": 1388524740000,
  "value": 265
}]
```

JSON with `rollup=false`

```
"metricValues": [
  {
    "current": 14,
    "max": 10024,
    "min": 2,
    "startTimeInMillis": 1388513700000,
    "value": 510
  },
  {
    "current": 14,
    "max": 10042,
    "min": 13,
    "startTimeInMillis": 1388513760000,
    "value": 327
  },
  {
    "current": 14,
    "max": 10008,
    "min": 13,
    "startTimeInMillis": 1388513820000,
    "value": 483
  },
  {
    "current": 13,
    "max": 10008,
    "min": 13,
    "startTimeInMillis": 1388513880000,
    "value": 854
  },
  {
    "current": 13,
    "max": 10052,
    "min": 13,
    "startTimeInMillis": 1388513940000,
    "value": 477
  },
  . . .
  . . .
]
```

Retrieve metric hierarchy

URI: /controller/rest/applications/<application-name| application-id>/metrics

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
----------------	----------------	-------	-----------

<application-name application-id>	URI	Provide either the application name or application id.	Yes
metric-path	Query	Provide the metric expression to get the metric data. The metric expression can fetch any elements visible in the metric browser. Use the pipe character to separate the parent and child name elements in the tree. Note: The pipe character must not appear at the beginning or at the end of the metric expression.	Yes
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve the metric hierarchy for the ACME Book Store.

URI	Sample object output
/controller/rest/applications/ACME Book Store Application/metrics?metric-path=BusinessTransaction Performance Business Transactions ECommerce Server ViewCart.sendItems	XML
/controller/rest/applications/ACME Book Store Application/metrics?metric-path=BusinessTransaction Performance Business Transactions ECommerce Server ViewCart.sendItems&output=JSON	JSON

Use Wild cards in Metric-Path Parameter

You can use the asterisk (*) wild card in the metric data to request metric data for all the instances of AppDynamics entity, such as a business transaction name, tier name or node name, in the metric path.

Example - Retrieve the average response time for all the tiers in the application using the wild card character for the tier name.

URI	Sample object output
<pre>/controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Overall Application Performance * Average Response Time (ms)&time-range-type=BEFORE_NOW&duration-in-mins=15</pre>	XML
<pre>/controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Overall Application Performance * Average Response Time (ms)&time-range-type=BEFORE_NOW&duration-in-mins=15&output=JSON</pre>	JSON

Example - Retrieve the CPU %Busy metric for all the nodes in the ECommerce tier using the wild card character for the node name.

URI	Sample object output
<pre>/controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Application Infrastructure Performance ECommerce Server Individual Nodes * Hardware Resources CPU %Busy&time-range-type=BEFORE_NOW&duration-in-mins=15</pre>	XML

<pre> /controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Application Infrastructure Performance ECommerce Server Individual Nodes * Hardware Resources CPU %Busy&time-range-type=BEFORE_NOW&duration-in-mins=15 &output=JSON </pre>	JSON
--	------

Example - Retrieve the CPU %Busy metric for all the nodes in all the tiers using wild card characters for the tier and node names.

URI	Sample object output
<pre> /controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Application Infrastructure Performance * Individual Nodes * Hardware Resources CPU %Busy&time-range-type=BEFORE_NOW&duration-in-mins=15 </pre>	XML
<pre> /controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Application Infrastructure Performance * Individual Nodes * Hardware Resources CPU %Busy&time-range-type=BEFORE_NOW&duration-in-mins=15 &output=JSON </pre>	JSON

Example - Retrieve the Calls per Minute metric for all the business transactions on the ECommerce tier using the wild card character for the business transaction name

URI	Sample object output
-----	----------------------

<pre> /controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Business Transaction Performance Business Transactions ECommerce Server * Calls per Minute&time-range-type=BEFORE_NOW &duration-in-mins=15 </pre>	XML
<pre> /controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Business Transaction Performance Business Transactions ECommerce Server * Calls per Minute&time-range-type=BEFORE_NOW &duration-in-mins=15&output=JSON </pre>	JSON

Retrieve metrics for a time range

You can fetch the data for any specified metrics for any time range.

Note that metric data REST URIs restrict the amount of data that can be returned. The maximum is 200 metrics.

URI: /controller/rest/applications/<application-name>/metric-data

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes

metric-path	Query	<p>The metric expression to get the metric data.</p> <p>Use the pipe character to separate the parent and child name elements in the tree.</p> <p>The Pipe delimiter must not appear at the beginning or at the end of the metric expression.</p>	Yes
-------------	-------	---	-----

time-range-type	Query	<p>Possible values are:</p> <p>BEFORE_NOW To use the "BEFORE_NOW" option, you must also specify the "duration-in-mins" parameter.</p> <p>BEFORE_TIME To use the "BEFORE_TIME" option, you must also specify the "duration-in-mins" and "end-time" parameters.</p> <p>AFTER_TIME To use the "AFTER_TIME" option, you must also specify the "duration-in-mins" and "start-time" parameters.</p> <p>BETWEEN_TIMES To use the "BETWEEN_TIMES" option, you must also specify the "start-time" and "end-time" parameters. The "BETWEEN_TIMES" range includes the start- time and excludes the end-time.</p>	Yes
duration-in-mins	Query	Duration (in minutes) to return the metric data.	If time-range-type is BEFORE_NOW , BEFORE_TIME , or AFTER_TIME
start-time	Query	Start time (in milliseconds) from which the metric data is returned in UNIX epoch time.	If time-range-type is AFTER_TIME or BETWEEN_TIMES

end-time	Query	End time (in milliseconds) until which the metric data is returned in UNIX epoch time.	If time-range-type is BEFORE_TIME or BETWEEN_TIMES
rollup	Query	Default is true. If true, value as single data point is returned. If false, all the values within the specified time range are returned.	No
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve the average response time for the past 15 minutes on the ECommerce server.

URI	Sample object output
/controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Overall Application Performance ECommerce Server Average Response Time (ms)&time-range-type=BEFORE_NOW&duration-in-mins=15	XML
/controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Overall Application Performance ECommerce Server Average Response Time (ms)&time-range-type=BEFORE_NOW&duration-in-mins=15&output=JSON	JSON

Example - Retrieve the multiple metrics, for the past 15 minutes, for the ViewCart.sendItems transaction on the ECommerce server.

URI	Sample object output
-----	----------------------

/controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Business Transaction Performance Business Transactions ECommerce Server ViewCart.sendItems *&time-range-type=BEFORE_NOW&duration-in-mins=15	XML
/controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Business Transaction Performance Business Transactions ECommerce Server ViewCart.sendItems *&time-range-type=BEFORE_NOW&duration-in-mins=15	JSON

Example - Retrieve the average cpu used by the All Other Traffic business transaction during the past 15 minutes on the ECommerce server.

Enter "APPDYNAMICS_DEFAULT_TX" for the All Other Traffic" transaction for the tier. See [All Other Traffic Business Transaction](#) for general information about the All Other Traffic business transaction.

URI	Sample object output
/controller/rest/applications/ACME Book Store Application/metric-data?metric-path=Business Transaction Performance Business Transactions ECommerce Server APPDYNAMICS_DEFAULT_TX Average CPU Used (ms)&time-range-type=BEFORE_NOW&duration-in-mins=15	XML

Example - Retrieve snapshots for the 15 minutes after Mon, 13 Aug 2012 08:20:41 for the ACME Book Store Application.

URI	Sample object output
/controller/rest/applications/3/request-snapshots?time-range-type=AFTER_TIME&start-time=1344846041495&duration-in-mins=15	XML
/controller/rest/applications/3/request-snapshots?time-range-type=AFTER_TIME&start-time=1344846041495&duration-in-mins=15&output=JSON	JSON

Example - Retrieve snapshots for the 15 minutes before Mon, 13 Aug 2012 08:20:41 for the ACME Book Store Application.

URI	Sample object output
-----	----------------------

/controller/rest/applications/3/request-snapshots?time-range-type=BEFORE_TIME&end-time=1344846041495&duration-in-mins=15	XML
/controller/rest/applications/3/request-snapshots?time-range-type=BEFORE_TIME&end-time=1344846041495&duration-in-mins=15&output=JSON	JSON

Example - Retrieve snapshots for the time range between Mon, 13 Aug 2012 08:20:41 and Mon, 13 Aug 2012 08:22:21 GMT for the ACME Book Store Application.

URI	Sample object output
/controller/rest/applications/3/request-snapshots?time-range-type=BEFORE_TIME&end-time=1344846041495&duration-in-mins=15	XML
/controller/rest/applications/3/request-snapshots?time-range-type=BEFORE_TIME&end-time=1344846041495&duration-in-mins=15&output=JSON	JSON

Retrieve all health rule violations in a particular business application

URI: /controller/rest/applications/<application-name|application-id>/problems/healthrule-violations

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes

time-range-type	Query	<p>Possible values are:</p> <p>BEFORE_NOW To use the "BEFORE_NOW" option, you must also specify the "duration-in-mins" parameter.</p> <p>BEFORE_TIME To use the "BEFORE_TIME" option, you must also specify the "duration-in-mins" and "end-time" parameters.</p> <p>AFTER_TIME To use the "AFTER_TIME" option, you must also specify the "duration-in-mins" and "start-time" parameters.</p> <p>BETWEEN_TIMES To use the "BETWEEN_TIMES" option, you must also specify the "start-time" and "end-time" parameters. The "BETWEEN_TIMES" range includes the start- time and excludes the end-time.</p>	Yes
duration-in-mins	Query	Duration (in minutes) to return the metric data.	If time-range-type is BEFORE_NOW , BEFORE_TIME , or AFTER_TIME
start-time	Query	Start time (in milliseconds) from which the metric data is returned.	If time-range-type is AFTER_TIME or BETWEEN_TIMES

end-time	Query	End time (in milliseconds) until which the metric data is returned.	If time-range-type is BEFORE_TIME or BETWEEN_TIMES
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Retrieve all policy violations in a particular business application

This URI is maintained for compatibility with pre-3.7 versions. Use [Retrieve all health rule violations in a particular business application](#) for 3.7 and later.

URI: /controller/rest/applications/<application-name|application-id>/problems/policy-violations

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes

time-range-type	Query	<p>Possible values are:</p> <p>BEFORE_NOW To use the "BEFORE_NOW" option, you must also specify the "duration-in-mins" parameter.</p> <p>BEFORE_TIME To use the "BEFORE_TIME" option, you must also specify the "duration-in-mins" and "end-time" parameters.</p> <p>AFTER_TIME To use the "AFTER_TIME" option, you must also specify the "duration-in-mins" and "start-time" parameters.</p> <p>BETWEEN_TIMES To use the "BETWEEN_TIMES" option, you must also specify the "start-time" and "end-time" parameters. The "BETWEEN_TIMES" range includes the start- time and excludes the end-time.</p>	Yes
duration-in-mins	Query	Duration (in minutes) to return the metric data.	If time-range-type is BEFORE_NOW , BEFORE_TIME , or AFTER_TIME
start-time	Query	Start time (in milliseconds) from which the metric data is returned.	If time-range-type is AFTER_TIME or BETWEEN_TIMES

end-time	Query	End time (in milliseconds) until which the metric data is returned.	If time-range-type is BEFORE_TIME or BETWEEN_TIMES
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve the list of all policy violations in the ACME Book Store for the past hour.

URI	Sample object output
/controller/rest/applications/ACME Book Store Application/problems/policy-violations?time-range-type=BEFORE_NOW&duration-in-mins=60&output=XML	XML
/controller/rest/applications/ACME Book Store Application/problems/policy-violations?time-range-type=BEFORE_NOW&duration-in-mins=60&output=JSON	JSON

Retrieve event data

You can capture data for the event types listed in the event-types parameter.

URI: /controller/rest/applications/<application-name|application-id>/events

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes

time-range-type	Query	<p>Possible values are:</p> <p>BEFORE_NOW To use the "BEFORE_NOW" option, you must also specify the "duration-in-mins" parameter.</p> <p>BEFORE_TIME To use the "BEFORE_TIME" option, you must also specify the "duration-in-mins" and "end-time" parameters.</p> <p>AFTER_TIME To use the "AFTER_TIME" option, you must also specify the "duration-in-mins" and "start-time" parameters.</p> <p>BETWEEN_TIMES To use the "BETWEEN_TIMES" option, you must also specify the "start-time" and "end-time" parameters. The "BETWEEN_TIMES" range includes the start- time and excludes the end-time.</p>	Yes
duration-in-mins	Query	Specify the duration (in minutes) to return the metric data.	If time-range-type is BEFORE_NOW , BEFORE_TIME , or AFTER_TIME
start-time	Query	Specify the start time (in milliseconds) from which the metric data is returned.	If time-range-type is AFTER_TIME or BETWEEN_TIMES

end-time	Query	Specify the end time (in milliseconds) until which the metric data is returned.	If time-range-type is BEFORE_TIME or BETWEEN_TIMES
event-types	Query	Specify the comma-separated list of event types for which you want to retrieve event information. See the Events Reference for the valid event types.	Yes
severities	Query	Specify the comma-separated list of severities for which you want to retrieve event information. The severities are: <ul style="list-style-type: none"> • INFO • WARN • ERROR 	Yes
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve the list of events of type "APPLICATION_ERROR" or "DIAGNOSTIC_SESSION" of any severity that occurred in the specified time range.

URI	Sample object output
/controller/rest/applications/ACME%20Book%20Store%20Application/events?time-range-type=BEFORE_NOW&duration-in-mins=30&event-types=%20APPLICATION_ERROR,DIAGNOSTIC_SESSION&severities=INFO,WARN,ERROR&output=XML	XML

```
/controller/rest/applications/ACME%20Book%
20Store%20Application/events?time-range-ty
e=BEFORE_NOW&duration-in-mins=30&even
t-types=%20APPLICATION_ERROR,DIAGNO
STIC_SESSION&severities=INFO,WARN,ER
ROR&output=JSON
```

JSON

Create Events

You can create APPLICATION_DEPLOYMENT and CUSTOM events.

Application Deployment Event Integration

The AppDynamics REST API lets you integrate events of type "APPLICATION_DEPLOYMENT" with other systems.

For example, suppose you want to create an event automatically in your AppDynamics monitored system for every new release. To integrate these systems, use the following REST API to create an event of type "APPLICATION_DEPLOYMENT" in your managed environment.

This is a POST request. You should receive the event ID after successful invocation of the request.

URI: /controller/rest/applications/<application-name|application-id>/events

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either application name or application id.	Yes
summary	Query	Provide the summary for the event.	Yes
comment	Query	Provide the comments (if any) for the event.	Yes
eventtype	Query	APPLICATION_DEPLOYMENT	Yes

Create a Custom Event

The AppDynamics REST API lets you create a custom event.

This is a POST request. You should receive the event ID after successful invocation of the request.

URI: /controller/rest/applications/<application-name|application-id>/events

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either application name or application id.	Yes
summary	Query	Provide the summary for the event.	No
comment	Query	Provide the comments for the event.	Yes
eventtype	Query	CUSTOM	Yes

Retrieve transaction snapshots for a Business Transaction for a time range

URI: /controller/rest/applications/<application-id>/request-snapshots

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-id>	URI	Provide either the application name or application id.	Yes

time-range-type	Query	<p>Possible values are:</p> <p>BEFORE_NOW</p> <p>To use the "BEFORE_NOW" option, you must also specify the "duration-in-mins" parameter.</p> <p>BEFORE_TIME</p> <p>To use the "BEFORE_TIME" option, you must also specify the "duration-in-mins" and "end-time" parameters.</p> <p>AFTER_TIME</p> <p>To use the "AFTER_TIME" option, you must also specify the "duration-in-mins" and "start-time" parameters.</p> <p>BETWEEN_TIMES</p> <p>To use the "BETWEEN_TIMES" option, you must also specify the "start-time" and "end-time" parameters. The "BETWEEN_TIMES" range includes the start-time and excludes the end-time.</p>	Yes
-----------------	-------	---	-----

duration-in-mins	Query	Duration (in minutes) to return the data.	If time-range-type is BEFORE_NOW , BEFORE_TIME , or AFTER_TIME
start-time	Query	Start time (in milliseconds) from which the data is returned.	If time-range-type is AFTER_TIME or BETWEEN_TIMES
end-time	Query	End time (in milliseconds) until which the data is returned.	If time-range-type is BEFORE_TIME or BETWEEN_TIMES
guids	Query	Array of comma-separated guids for the transaction snapshots. If not specified, retrieves all snapshots in the specified time range.	No
archived	Query	True to retrieve archived snapshots. Default is false.	No

deep-dive-policy	Query	<p>Array of comma-separated snapshot policy filters to apply. Valid values are:</p> <ul style="list-style-type: none">• SLA_FAILURE• TIME_SAMPLING• ERROR_SAMPLING• OCCURRENCE_SAMPLING• ON_DEMAND• HOTSPOT• HOTSPOT_LEARN• APPLICATION_STARTUP• SLOW_DIAGNOSTIC_SESSION• ERROR_DIAGNOSTIC_SESSION• POLICY_FAILURE_DIAGNOSTIC_SESSION• DIAGNOSTIC_SESSION• INFLIGHT_SLOW_SESSION	No
application-component-ids	Query	<p>Array of comma-separated tier IDs to filter. Default is all the tiers in the application.</p>	No

application-component-node-ids	Query	Array of comma-separated node ID filters. Default is all the nodes in the application	No
business-transaction-ids	Query	Array of comma-separated business transaction ID filters. Default is all the business transactions in the application.	No
user-experience	Query	Array of comma-separated user experiences filters. Valid values are: <ul style="list-style-type: none">• NORMAL• SLOW• VERY_SLOW• STALL• ERROR	No
first-in-chain	Query	If true, retrieve only the first request from the chain. Default is false.	No

need-props	Query	<p>If true, the values of the following snapshot properties are included in the output. These values correspond to the values of the data-collector-type parameter. If false, these values are empty in the REST output. The default is false. <i>New in 3.7.11.</i></p> <ul style="list-style-type: none">• errorDetails• errorIDs• httpParameters• businessData• cookies• httpHeaders• sessionKeys• responseHeaders• logMessages• transactionProperties• transactionEvents• dotnetProperty	No	
need-exit-calls	Query	<p>If true, exit calls are included in the result. Default is false.</p>	No	
execution-time-in-milis	Query	<p>If set, retrieves only data for requests with execution times greater than this value.</p>	No	

session-id	Query	If set, retrieves data only for this session id.	No
user-principal-id	Query	If set, retrieves data only for this user login.	No
error-ids	Query	Array of comma-separated error codes to filter by. Default is to retrieve all error codes.	No
starting-request-id, ending-request-id	Query	If set, retrieves data only for this range of request IDs.	No
error-occurred	Query	If true, retrieves only error requests. Default is false.	No
diagnostic-snapshot	Query	If true, retrieves only diagnostic snapshots. Default is false.	No
bad-request	Query	If true, retrieves only slow and error requests. Default is false.	No
diagnostic-session-guid	Query	Array of comma-separated diagnostic session guides to filter.	No
data-collector-name	Query	Used with data-collector-value to filter snapshot collection based on the value of a data collector. <i>New in 3.6.1.</i>	No

data-collector-value	Query	Used with data-collector-name to filter snapshot collection based on the value of a data collector. <i>New in 3.6.1.</i>	If data-collector-name is set.
----------------------	-------	--	--------------------------------

data-collector-type	Query	<p>Used with data-collector-name and data-collector-value to filter snapshot collection based on the value of a data collector. <i>New in 3.6.1.</i> Some of the values contain spaces. All are case-sensitive and where indicated the spaces are required. Valid values are:</p> <ul style="list-style-type: none"> • Error IDs • Stack Traces • Error Detail • Http Parameter • Business Data (This type is a method invocation data collector.) • Cookie • Http Header • Session Key • Response Header • Log Message • Transaction Property • Transaction Event • Dotnet Property • isProtoBuf • EUM Request GUID
---------------------	-------	---

Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No
--------	-------	---	----

Example - Retrieve list of transaction snapshots for the ACME Book Store.

URI	Sample object output
/controller/rest/applications/AcmeOnlineBookStore/request-snapshots?time-range-type=BEFORE_NOW&duration-in-mins=2	XML
/controller/rest/applications/AcmeOnlineBookStore/request-snapshots?time-range-type=BEFORE_NOW&duration-in-mins=2&output=JSON	JSON

Example - Retrieve list of transaction snapshots including the snapshot fields that are associated with an Http Parameter data collector.

URI	Sample object output
controller/rest/applications/2/request-snapshots?time-range-type=BEFORE_NOW&duration-in-mins=120&data-collector-type=Http%20Parameter&data-collector-name=param1&data-collector-value=%5B100%5D&need-props=true]	XML
controller/rest/applications/2/request-snapshots?time-range-type=BEFORE_NOW&duration-in-mins=120&data-collector-type=Http%20Parameter&data-collector-name=param1&data-collector-value=%5B100%5D&need-props=true&output=JSON]	JSON

Create and modify AppDynamics users

This is an HTTP POST operation.

The create and modify user URIs are identical except for the user-id parameter, which is not passed for the create operation. The user-id is generated by the create operation. A response code of 200 indicates success.

URI: /controller/rest/users

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
user-name	Query	user name	Yes
user-id	Query	user id	No for a create; yes for an update
user-display-name	Query	display name	Yes
user-roles	Query	comma-separated list of roles	No
user-password	Query	user password	Yes for create; optional for update
user-email	Query	user email	Yes

Include or exclude a business transaction from monitoring

This is an HTTP POST operation.

To exclude a business transaction from monitoring, set the exclude parameter to true.

To turn on monitoring for a currently excluded business transaction, set the exclude parameter to false.

Send the list of business transactions to be excluded or re-included as xml payload, not as parameters.

A sample business-transaction-list is:

```
<business-transactions>
  <business-transaction>
    <id>15</id>
  </business-transaction>
  <business-transaction>
    <id>16</id>
  </business-transaction>
</business-transactions>
```

Make sure the Content-Type header is set to "application/xml".

URI: /controller/rest/applications/<application-name | application-id>/business-transactions

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes

exclude	Post	true false	Yes
---------	------	------------	-----

Example - Exclude business transaction 166 from monitoring.

URI
/controller/rest/applications/AcmeOnlineBookStore/business-transactions?exclude=true

This is the way the request appears in the Google Advanced REST Client:

Request type is POST.

Business transaction list goes here.

Set content type to application/xml.

application/xml

Status: 200 OK Loading time: 224 ms

Retrieve all controller global configuration values

These are the values that you set interactively from the Controller Settings screen in the AppDynamics Administration console.

Access requires the "root" password, which was created for the AppDynamics admin user when the controller was installed. See [Access the Administration Console](#).

URI: /configuration

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
----------------	----------------	-------	-----------

Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No
--------	-------	---	----

Example - Retrieve list of all global configuration values

URI	Sample object output
/controller/rest/configuration	XML
/controller/rest/configuration?output=JSON	JSON

Retrieve a single controller global configuration value

URI: /configuration?name=<controller_setting_name>

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
name	Query	Name of the Controller setting to retrieve	Yes
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve the global metrics buffer size.

URI	Sample object output
/controller/rest/configuration?name=metrics.buffer.size	XML
/controller/rest/configuration?name=metrics.buffer.size&output=JSON	JSON

Configure Global Controller Settings

This is an HTTP POST operation.

URI: /controller/rest/configuration

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
name	Query	name of a controller setting; to see all the names, retrieve all the values as described in Example - Retrieve list of all global configuration values	Yes
value	Query	value to set	Yes

Mark Nodes as Historical

This is an HTTP POST operation.

Pass as parameters a comma-separated list of one or more node ids of the nodes to be marked as historical. You can specify up to a maximum of 25 node ids to be marked.

AppDynamics stops collecting metrics for a node that is marked as historical.

By default AppDynamics marks as historical (soft deletes) a node that has lost contact with the Controller for the number of hours configured in the node.retention.period controller setting. The default is 500 hours.

Information from a historical node can be retrieved until the node has lost contact with the controller for the number of hours configured in the node.permanent.deletion.period, after which time the historical node is permanently deleted. The default is 720 hours.

URI: /controller/rest/mark-nodes-historical?application-component-node-ids=44,45

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
application-component-node-ids	Query	comma-separated list of node ids	Yes

Example - Mark nodes 44 and 45 as historical.

URI	Sample object output
/controller/rest/mark-nodes-historical?application-component-node-ids=44,45	XML

Retrieve all policy violations in a particular business application

URI: /controller/rest/applications/<application-name|application-id>/problems/policy-violations

Input parameters

Parameter Name	Parameter Type	Value	Mandatory
<application-name application-id>	URI	Provide either the application name or application id.	Yes
time-range-type	Query	<p>Possible values are:</p> <p>BEFORE_NOW To use the "BEFORE_NOW" option, you must also specify the "duration-in-mins" parameter.</p> <p>BEFORE_TIME To use the "BEFORE_TIME" option, you must also specify the "duration-in-mins" and "end-time" parameters.</p> <p>AFTER_TIME To use the "AFTER_TIME" option, you must also specify the "duration-in-mins" and "start-time" parameters.</p> <p>BETWEEN_TIMES To use the "BETWEEN_TIMES" option, you must also specify the "start-time" and "end-time" parameters.</p>	Yes
duration-in-mins	Query	Duration (in minutes) to return the metric data.	If time-range-type is BEFORE_NOW , BEFORE_TIME , or AFTER_TIME

start-time	Query	Start time (in milliseconds) from which the metric data is returned.	If time-range-type is AFTER_TIME or BETWEEN_TIMES
end-time	Query	End time (in milliseconds) until which the metric data is returned.	If time-range-type is BEFORE_TIME or BETWEEN_TIMES
incident-status	Query	If OPEN, retrieve only the policy violations that are open. If RESOLVED, retrieve only the policy violations that are resolved.	No
Output	Query	HTTP Request parameter included as part of the URL to change the output format. Valid values are "XML" (default) or "JSON".	No

Example - Retrieve the list of all open policy violations in the ACME Book Store for the past 1000 minutes.

URI	Sample object output
/controller/rest/applications/2/problems/policy-violations?time-range-type=BEFORE_NOW&duration-in-mins=1000&incident-status=OPEN	XML
/controller/rest/applications/2/problems/policy-violations?time-range-type=BEFORE_NOW&duration-in-mins=1000&incident-status=OPEN&output=JSON	JSON

Example Integrations Using the AppDynamics REST API

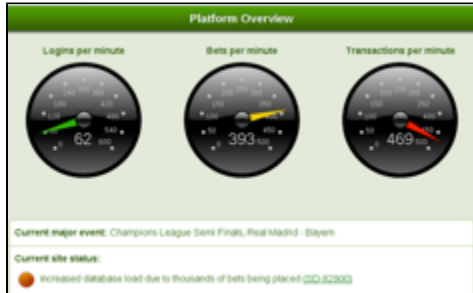
- [Python Client for the REST API](#)
- [Mobile Application](#)
- [Jenkins Plugin](#)

Python Client for the REST API

[REST Python Client Extension](#)

Mobile Application

See [Going Mobile with AppDynamics REST API](#) for information about a monitoring application that runs on a smart phone.



The code is available at: <https://github.com/unixunion/appdyngauges>

Courtesy Kegan Holtzhausen, thank you!

Jenkins Plugin

This plugin makes it possible to integrate data from AppDynamics into your Jenkins build.

The plugin is available at: <https://wiki.jenkins-ci.org/display/JENKINS/AppDynamics+Plugin>

Courtesy Miel Donkers, thank you!

Use Curl to Access AppDynamics REST APIs

You can fetch the contents of the AppDynamics REST API calls using curl. curl dumps the XML results to standard output.

The syntax is:

curl --user <appdynamics-username>@<appdynamics-account-name:<appdynamics-password> ' <URI>'

For example, this command gets the applications on the "demo" server:

```
curl --user user1@customer1:secret  
'http://demo.appdynamics.com/controller/rest/applications'
```

This command gets the tiers in application 2:

```
curl --user user1@customer1:secret  
'http://demo.appdynamics.com/controller/rest/applications/2/tiers'
```

This command gets the average response time in the Ecommerce tier of the ACME Online Book Store application over the past two hours:

```
curl --user user1@customer1:secret
'http://demo.appdynamics.com/controller/rest/applications/Acme%20Online%20Book%20Store/metric-data?metric-path=Business%20Transaction%20Performance%7CBusiness%20Transactions%7CE-Commerce-2%7CCheckout%7CAverage%20Response%20Time%20(ms)&time-range-type=BEFORE_NOW&duration-in-mins=120'
```

Integrate with AppDynamics for Databases

- [Prerequisites for AppDynamics for Databases Integration](#)
- [Configuring AppDynamics to Interface with AppDynamics for Databases](#)
 - [To Configure AppDynamics to Interface with AppDynamics for Databases](#)
 - [To Configure the App Agent for Java to Interface with the AppDynamics for Databases Oracle Collector](#)

This topic describes how you can link to AppDynamics for Databases (AppD4DB) from a monitored database, server, or storage system (NetApp) that is discovered by AppDynamics and supported by AppD4DB. This integration provides access to the deep database performance metrics provided by AppD4DB.

Prerequisites for AppDynamics for Databases Integration

- To use this integration you must have an AppD4DB license.
- AppDynamics for Databases must be configured to monitor the databases that you want to link to from AppDynamics.
- In addition, if you are installing AppD4DB and the AppDynamics controller on the same host, you may need to change the port number AppD4DB uses because the default port number for both products is 8090. We recommend that you use port 9090. For information about changing the port number on AppD4DB, see [Install AppD4DB and AppD Controller on the Same Host](#).

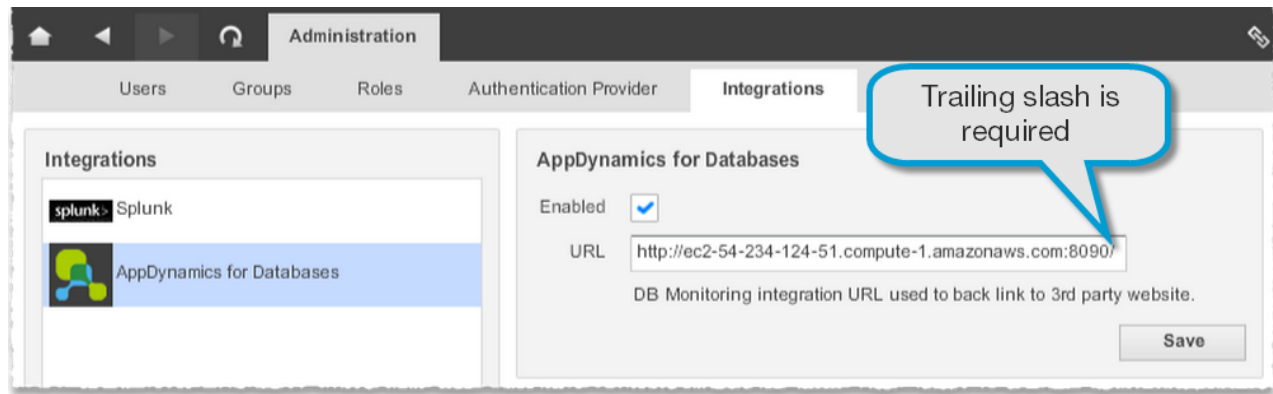
See [Linking to AppDynamics for Databases](#), and [Integrate and Use AppD4DB with AppDynamics Pro](#) for information on how to access AppD4DB from the Controller UI.

See [the AppDynamics for Databases documentation](#) for information about using AppD4DB.

Configuring AppDynamics to Interface with AppDynamics for Databases

To Configure AppDynamics to Interface with AppDynamics for Databases

1. In the upper right corner of the menu bar, click **Settings**.
2. From the dropdown menu, click **Administration**.
3. Click the **Integration** tab.
4. Click AppDynamics for Databases in the Integrations list.



5. Check the **Enabled** check box to enable the integration.

6. Enter the URL of your AppDynamics for Databases installation.

Note: The trailing slash in the URL is required.

7. Click **Save**.

To Configure the App Agent for Java to Interface with the AppDynamics for Databases Oracle Collector

New in AppDynamics for Databases 2.8 and AppDynamics 3.8 From a Transaction Snapshot Flow Map where the exit call is to an Oracle database, you can link to AppDynamics for Databases to see and analyze the exact SQL that was running at the time of the transaction snapshot. To enable this functionality, for the node containing the Oracle database, you must set the App Agent for Java node property `jdbc-dbcam-integration-enabled=true`.

Configure Integrations

- To configure integrations

You can enable and perform basic configuration with external products from the Administration window.

To configure integrations

1. As an administrator in the Controller UI, click **Settings** -> **Administration**.
2. Click the **Integration** tab. If the tab doesn't appear, click the right arrows to view additional tabs.
3. In the left panel select the product for which you want to configure integration.
The fields specific to the selected product appear in the right panel.
4. Check the Enabled check box to enable the integration. Clear this check box to disable the integration.
5. Provide the product-specific information in the text fields.
6. Click **Save**.

Integrate AppDynamics with Apica

AppDynamics integrates with:

- Apica WebPerformance
- Apica ProxySniffer
- Apica LoadTest

Apica users can drill down from these Apica products into the AppDynamics console to investigate the root cause of performance problems.

For more information about the integration with Apica see [the AppDynamics blog](#) and [the Apica-AppDynamics partner page](#).

Integrate AppDynamics with BMC End User Experience Management

- [To Integrate AppDynamics with BMC End User Experience Management](#)
 - [Setting Up AppDynamics](#)
 - [Setting up BMC End User Experience Management](#)
 - [Verifying the Integration](#)
 - [Check the status of the Application Visibility servers configured](#)
 - [Check the reference list for the "Trace \(object-level\)" custom field](#)
- [User Accounts for Controller SaaS Users](#)
- [Troubleshooting BMC EUEM-AppDynamics Integration](#)
 - [Do you have an enterprise level license?](#)
 - [Is the Controller available and can it return its status?](#)
 - [Is the BMC EUEM visibility server configuration correct?](#)
 - [Is the controller.services.hostName address in the Controller domain.xml file correct?](#)
 - [Is the AppDynamics Administration configuration correct?](#)
 - [Was the Java App Agent started or restarted AFTER the Controller integration settings changed?](#)
 - [Is AppDynamics constructing the correct URLs to link to snapshots?](#)
 - [Is the BMC EUEM system seeing the required response headers from AppDynamics?](#)
 - [Is the BMC EUEM system reporting Snapshot status for pages in the Session browser?](#)
 - [Why are all the drilldown icons red?](#)
 - [Where is the log file containing information about Application Visibility events?](#)
- [Learn More](#)

This topic describes how to integrate AppDynamics with the BMC End User Experience Management (EUEM) appliance.

Note that this feature was previously known as Coradiant. See <http://www.bmc.com/products/brand/coradiant.html> for information about the integration with BMC.

To Integrate AppDynamics with BMC End User Experience Management

Setting Up AppDynamics

 An AppDynamics for BMC Software Solutions license is required.

Steps 1 through 3 are for on-premise installations. If you are using the AppDynamics SaaS Controller, skip to Step 4.

1. Install the AppDynamics Controller. For details see [Install the Controller on Linux](#) or [Install the](#)

Controller on Windows.

2. Stop the Controller. Go to the command line console and execute the following command:

For Windows:

```
controller.bat stop
```

For Linux:

```
./controller.sh stop
```

3. Open the AppDynamics Controller domain.xml file located at `<controller_install>/appdynamics-controller/appserver/glassfish/domains/domain1/config`. Configure the host name using the following JVM options:

```
<jvm-options>
-Dappdynamics.controller.services.hostName=<controller_host>
</jvm-options>
```

The `controller_host` is the IP address or domain (`www.host.com`) host that you configured during Controller installation. For details see [Install the Controller on Linux](#) or [Install the Controller on Windows](#).

For example, if the `controller_host` was configured to `192.10.10.1`, the `-D` property will have the following value:

```
<jvm-options>
-Dappdynamics.controller.services.hostName=192.10.10.1
</jvm-options>
```

The `controller.services.hostName` property is used to connect the BMC EUEM User Interface with the AppDynamics Controller. The URL points to a snapshot in the AppDynamics UI. The host name should be an IP or domain name that is accessible to the end users of the EUEM and AppDynamics user interfaces. If you need to change the `hostName` see [Changing the Host Name](#).

4. In the Controller domain.xml file, change the setting of the `appdynamics.controller.ui.deeplink.url` property. The property requires a full URL.

```
appdynamics.controller.ui.deeplink.url=http://<controller_host>:<port>/controlle
r
```

5. Save the domain.xml file.

6. Restart the Controller at a command line console:

For Windows:

```
controller.bat start
```

For Linux:

```
./controller.sh start
```

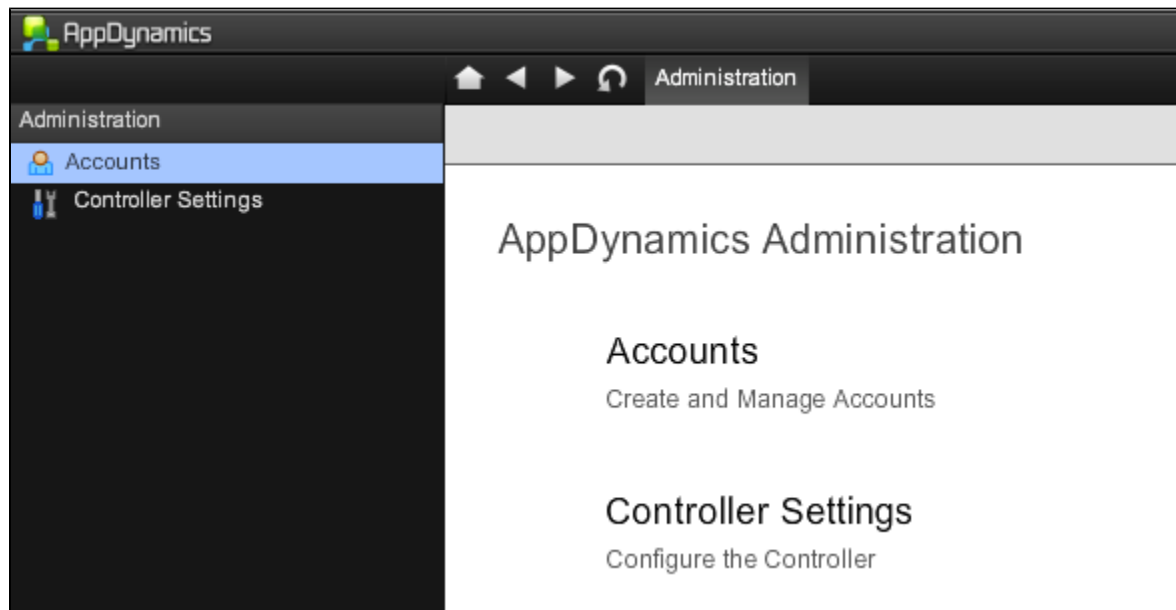
7. Configure the Integration Parameters.

From a browser, log in to the AppDynamics Admin UI as the Admin user:

```
http://<controller_host>:<port>/controller/admin.html
```

Note: The UI may not open the Accounts screen if the Controller is not in multi-tenant mode. To work around this, use the URL parameter enableAccounts=true.

```
http://<controller_host>:<port>/controller/admin.html?enableAccounts=true
```



8. Click **Controller Settings** to open the list of controller settings.

9. Set the vendor identifier to "ad" and click **Save**.

controller.vendor.identifier	Controller vendor id	ad	Save
------------------------------	----------------------	----	------

10. Set the server identifier and click **Save**.

A server identifier is required.

Set this property to "CONTROLLER_1".

controller.server.identifier	Controller server id	CONTROLLER_1	Save
------------------------------	----------------------	--------------	------

11. On the left navigation pane, Click **Accounts** to open the Accounts list.

12. Double-click the customer1 account row to open the customer1 account properties editor.

customer1	customer1	SJ5b2m7d1\$354	100000	100000	100000	No expiration date
-----------	-----------	----------------	--------	--------	--------	--------------------

13. Set the BMC EUEM integration properties. An AppDynamics for BMC Software Solutions license is required for these options to be available.

- Click the **BMC End User Experience Management Enabled** checkbox.
- Set the URL to the BMC EUEM Analyzer. Be sure to add the "/" character at the end of the URL.

BMC End User Experience Management Enabled	<input checked="" type="checkbox"/>
BMC End User Experience Management Sever URL	<input type="text" value="http://192.1.22.10/"/>
This is the URL to the BMC Server. For example http://server:port/. Note that you need the last / in there.	

Setting up BMC End User Experience Management

These instructions apply to Real User Analyzer or Real User Monitor.

1. Log in to BMC EUEM as the security officer.
2. Navigate to **Administration->Integration->Application Visibility servers**.
3. Add an Application Visibility server with the following parameters:

```
URI: http://<controller_host>:<port>/controller/rest
Authentication method: None
Username: (not needed)
Password: (not needed)
```

Verifying the Integration

The integration is successful when the response headers from AppDynamics are visible in the BMC EUEM UI. There are two ways to verify the integration:

#Check the status of the Application Visibility servers configured

Check the reference list for the "Trace (object-level)" custom field

Check the status of the Application Visibility servers configured

1. In the BMC End User Experience Management UI (Real User Monitor or Real User Analyzer),

navigate to the **Administration->Integration->Application Visibility servers** page.

2. Check whether the configured AppDynamics server has a green checkmark under the **Connecti on** column.

3. If there is a green checkmark, hover the mouse over the icon under the computer icon and verify the settings of the AppDynamics server.

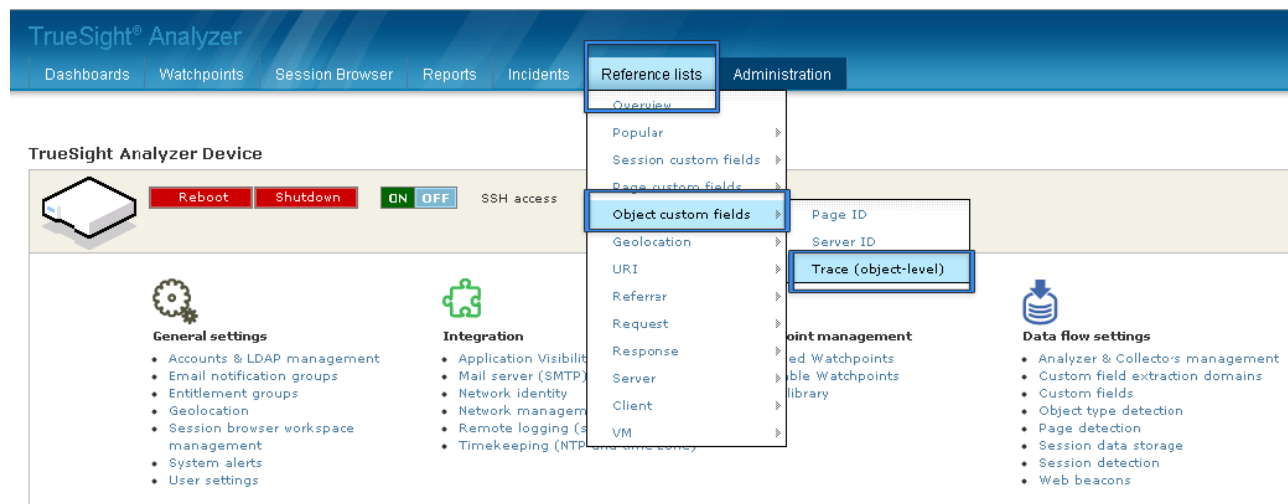
Check the reference list for the "Trace (object-level)" custom field

1. In the BMC End User Experience Management UI (Real User Monitor or Real User Analyzer), open the **Reference lists** menu.

2. Select **Object custom fields** then **Trace (object-level)**.

3. There should be a list of values for this custom field in the form of:

```
vid=ad&sid=CONTROLLER_1&tid=<GUID of a Business Transaction>
```



If the integration is successful, you will see a list of values captured by AppDynamics.

If you do not see any values for the Trace custom field, either the BMC EUEM system is not configured properly or no headers are being sent by the AppDynamics system. For assistance see [Troubleshooting BMC EUEM-AppDynamics Integration](#).

User Accounts for Controller SaaS Users

When integrating BMC EUEM with AppDynamics SaaS Controllers, the username and password must be provided during the integration.

For a multi-tenant Controller such as the AppDynamics SaaS Controller, the user name format is <user-name>@<account-name>. SaaS customers receive the user-name and account-name in the Welcome email sent by the AppDynamics Support Team.

Troubleshooting BMC EUEM-AppDynamics Integration

The following table summarizes the settings that are required to enable a successful BMC EUEM-AppDynamics integration.

Description	What to set	Required by AppDynamics	Required by BMC EUEM
Application Visibility server (AppDynamics Controller) URL	http://<controller_host>:<port>/controller/res t	No	Yes
Controller SaaS User credentials	<user-name>@<account-name>	No	Yes
Controller host name or IP address	controller.services.hostName in the domain.xml file	Yes	Yes
Controller port	default is 8090	Yes	Yes
Controller deep Link URL	appdynamics.controller.ui.deeplink.url in the domain.xml file	Yes	No
Controller vendor ID	controller.server.identifier property in the Controller Settings must be "ad"	Yes	No
Controller server name	controller.vendor.identifier property in the Controller Settings	Yes	No
Turn on EUEM in AppDynamics	the BMC End User Experience Management Enabled option in the Accounts Settings	Yes	No
URL pointing to a BMC EUEM Real User Analyzer or BMC EUEM Real User Monitor	The BMC End User Experience Management Server URL setting in the Account Settings	Yes	No

To determine where a problem may lie, check the following information.

Do you have an enterprise level license?

In the license, check to make sure that you have the following line:

```
property_edition=enterprise
```

Normally this line does not show, but it can be seen if you check "BMC integration" in Salesforce.

Is the Controller available and can it return its status?

Open the Controller URL in a browser that is in the same network segment as the BMC EUEM Real User Analyzer/Monitor:

```
http://<controller_host>:<port>/controller/rest/serverstatus
```

An XML file with AppDynamics metadata should display. If it does not, then the Controller is not available. If you are using the SaaS server, contact AppDynamics Support. If you are using an on-premise server, you may need to restart it.

Is the BMC EUEM visibility server configuration correct?

In the BMC EUEM UI, the AppDynamics URL should be:

```
http://<controller_host>:<port>/controller/rest
```

Is the controller.services.hostName address in the Controller domain.xml file correct?

The Controller services host name option should be set to your Controller server:

```
<jvm-options>
-Dappdynamics.controller.services.hostName=<controller_host>
</jvm-options>
```

The controller_host is the IP or domain address that accesses the Controller on your network. The Controller uses this address when it provides deep links to snapshots.

Is the AppDynamics Administration configuration correct?

For AppDynamics 3.3.3 and newer, in the **Administration -> Controller Settings** the controller.server.identifier property must be correct. When using the Controller SaaS service, it must be exactly set to:

```
CONTROLLER_1
```

Also the controller.vendor.identifier property must be exactly set to:

```
ad
```

⚠ Remember to **Save** each property!

Also check the **Administration->Accounts->Account Details** settings and confirm that:

- The **BMC End User Experience Management Enabled** option is checked
- The **BMC End User Experience Management Server URL** is the URL to the BMC server, including the slash "/" at the end of the URL

For AppDynamics 3.3.2 and older, use these settings:

```
coradiant.integration.url = https://ts02.coradiant.com/
```

The last character "/" is required.

```
coradiant.integration.enabled = true
```

⚠ Remember to save each property!

Was the Java App Agent started or restarted AFTER the Controller integration settings changed?

When the integration settings change in the Controller Admin UI, the Agent must restart for the changes to take effect.

Is AppDynamics constructing the correct URLs to link to snapshots?

Follow this procedure to get details about snapshots.


1. In AppDynamics, locate a snapshot with a call graph and copy the request GUID. In BMC EUEM this is called the trace ID (tid).
2. In a text file, construct a URL for the snapshot that includes the request tid/GUID:

```
http://<controller_host>:<port>/controller/rest/tracestatus?tid=<requestGuid>
```

3. Run the URL in a browser and see what is returned. For example:

```
<tracelist vendorid="" version="1"><trace
tid="855a90b1-5698-45c0-bded-22a6374981fc"><available>true</available>
<displayurl>http://111.0.0.1:8080/controller/#location=APP_SNAPSHOT_VIEWER&application=2&requestGUID=855a90b1-5698-45c0-bded-22a6374981fc</displayurl></trace></tracelist>
```

4. Copy the <displayurl> and run it in a browser. You should see the snapshot.

 If you receive a 500 error, check your license. You may not have an enterprise license.

Is the BMC EUEM system seeing the required response headers from AppDynamics?

The Application Visibility integration works by allowing you to navigate from a page view in the Session browser to a transaction on the AppDynamics system. The best way to look for pages with an AppDynamics link is to create a Page Watchpoint with the following filter expression:

```
NOT (trace is null)
```

From the Watchpoints page, use the Action button associated with this watchpoint to drill down to these pages and access the Session browser.

Is the BMC EUEM system reporting Snapshot status for pages in the Session browser?

Next to each page there can be four types of icons representing different statuses about the link from the page to the AppDynamics transaction. They are explained as follows.

- A clock icon indicates that the BMC EUEM system is trying to contact the AppDynamics system. If this icon persists, verify the communication between the BMC EUEM system and the AppDynamics system. Contact your IT administrator for assistance.
- A green icon indicates that the BMC EUEM system was able to contact the AppDynamics system and a Snapshot was found for that Business Transaction. Click the details link to navigate to the AppDynamics transaction.
- A yellow icon indicates that the BMC EUEM system was able to contact the AppDynamics system but the captured tid/GUID is no longer available. Contact your AppDynamics administrator for assistance about why the transaction no longer exists. In the BMC EUEM Real User Analyzer software prior to version 1.1 and BMC EUEM Real User Monitor software prior to version 5.1, when the BMC EUEM system contacts the AppDynamics system but the captured tid/GUID is no longer available, a yellow icon is displayed in the Session browser result window. More recent versions of BMC EUEM have suppressed the yellow icon and the behavior is replaced by the clock icon appearing then disappearing. Contact your AppDynamics administrator for assistance about why the transaction no longer exists. To re-enable the yellow icon functionality, please contact BMC Customer Support.
- A red icon indicates that the BMC EUEM system tried to contact the AppDynamics system but did not get a response for the tid/GUID captured. Click the page to see the page detail and click the Custom fields link. Verify that the Trace custom field has a value and try to search the value of the tid/GUID on the AppDynamics system using Advanced Search. If the search does not work, contact your AppDynamics Support representative for assistance.

Note: If you refresh the Session browser result window, all the Snapshot status will be rechecked for the pages shown. This is a good way to quickly check the new status after you've made some changes.

Why are all the drilldown icons red?

If the Session browser result window displays a lot of pages, a timeout may be reached when

requesting the status for all of them from the AppDynamics controller. By default this timeout is 12 seconds. If there is a firewall or network device that adds latency to the communication between BMC EUEM and AppDynamics systems then this timeout will need to be increased.

To increase the timeout contact a BMC Support engineer.

Where is the log file containing information about Application Visibility events?

1. Log into EUEM as the security officer and access this URL:

```
http://EUEM-host/tools/logs.do
```

2. Click the link for “Application Visibility events log”.

Learn More

- [AppDynamics EUM Integration](#)
- [BMC End User Experience Management](#)

Integrate AppDynamics with DB CAM

- [Prerequisites for DB CAM Integration](#)
- [Configuring AppDynamics to Interface with DB CAM](#)
 - [Configure DB CAM at the Account Level](#)
 - [To Configure One AppDynamics Account for DB CAM Integration](#)
 - [Configure DB CAM at the Agent Level](#)
- [Linking to DB CAM from AppDynamics](#)
 - [To Link to DB CAM from a Dashboard](#)
 - [To Link to DB CAM from a Transaction Snapshot](#)
- [Learn More](#)

You can link to DB CAM for any DB CAM-monitored database that is discovered by AppDynamics. This integration provides access to the database performance metrics provided by DB CAM.

Prerequisites for DB CAM Integration

To use this integration you must have a DB CAM license.

DB CAM must be configured to monitor the databases that you want to link to from AppDynamics.

Configuring AppDynamics to Interface with DB CAM

You configure DB CAM integration at the account level and at the app agent level.

Configure DB CAM at the Account Level

Configure the integration at the account level using the Administration Console at

```
<host>:<port>/controller/admin.html
```

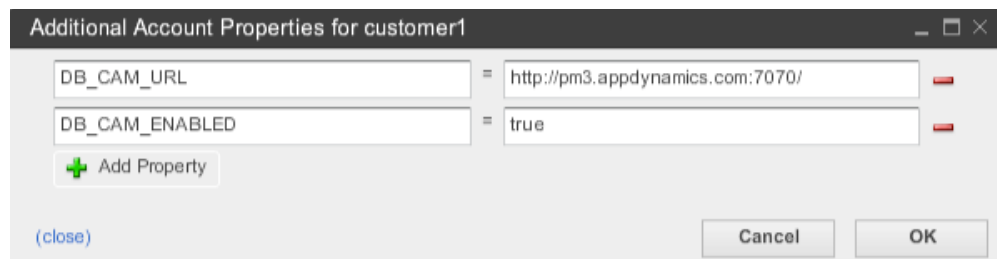
Create two new properties as name-value pairs in each account for which you want to enable DB CAM integration.

To Configure One AppDynamics Account for DB CAM Integration

1. Login to the Administrator Console with the administrator root password.
2. Select Accounts.
3. In the accounts list, double-click the account for which you want to configure DB CAM integration.
4. In the upper right corner of the account screen, click **Additional Account Properties**.
5. In the Additional Account Properties screen, click **Add Property** to add the DB_CAM_URL property.
6. In the left field enter "DB_CAM_URL".
7. In the right field enter the URL of the AppDynamics Controller that you are configuring using the syntax

```
http[s]://<host>:<port>
```

8. Click **Add Property** again to add the DB_CAM_ENABLED property.
9. In the left field enter "DB_CAM_ENABLED".
10. In the right field enter "true".
11. Click **OK** to save the properties.
12. Log out of the Administrator Console.



Configure DB CAM at the Agent Level

For each app agent for which you want to enable access to deep diagnostics from DB CAM:

1. Open the AppServerAgent/conf/app-agent-config.xml file for the app agent.
2. Locate the TransactionMonitoringService element:

```
<agent-service name="TransactionMonitoringService" enabled="true">
```

3. Add the jdbc-dbcam-integration-enabled property for the service:

```
<agent-service name="TransactionMonitoringService" enabled="true">

<service-dependencies>BCIEngine, SnapshotService</service-dependencie
s>

    <configuration-properties>
        <property name="jdbc-dbcam-integration-enabled"
value="true"/>
    </configuration-properties>
</agent-service>
```

4. Save the file.

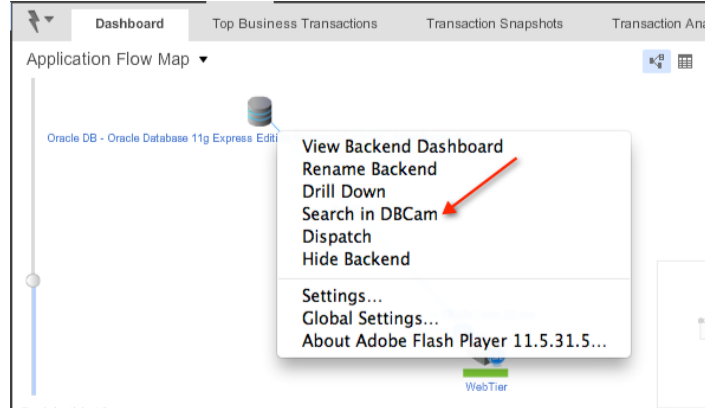
Linking to DB CAM from AppDynamics

You can link to DB CAM from any AppDynamics flow map that displays a discovered DB CAM-monitored database. The flow map could be in a dashboard or a transaction snapshot.

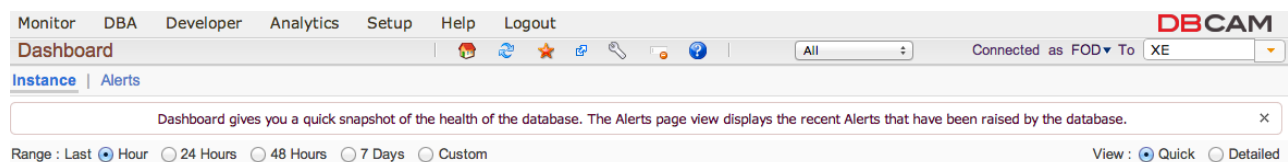
If you link to DB CAM from a dashboard you will land in the DB CAM instance dashboard. If you link to be DB CAM from a transaction snapshot, you will land in the DB CAM Session Drill Down screen.

To Link to DB CAM from a Dashboard

1. In the flow map of a dashboard, right-click on the link below a database icon.
2. Click **Search in DBCam**.

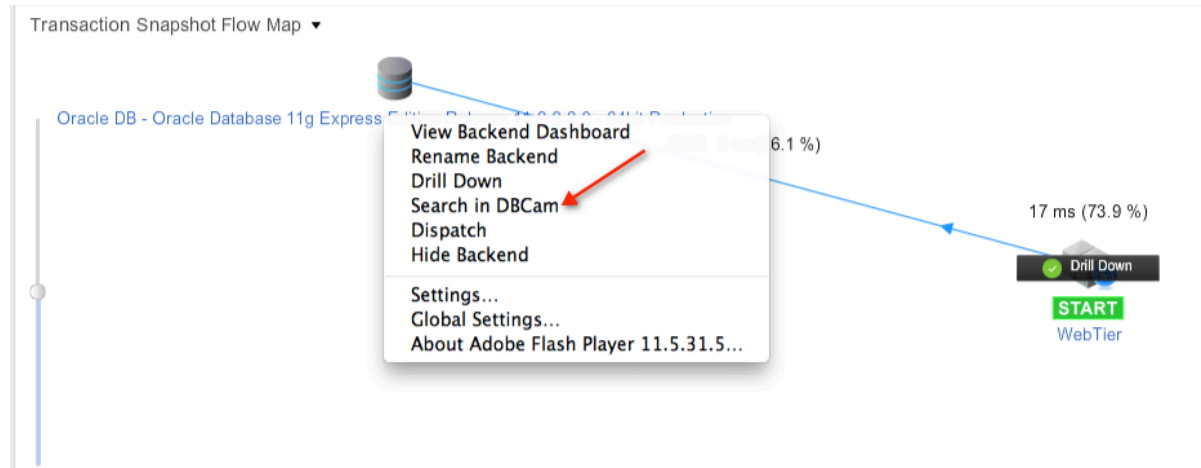


DB CAM launches and displays the instance dashboard for the selected database.

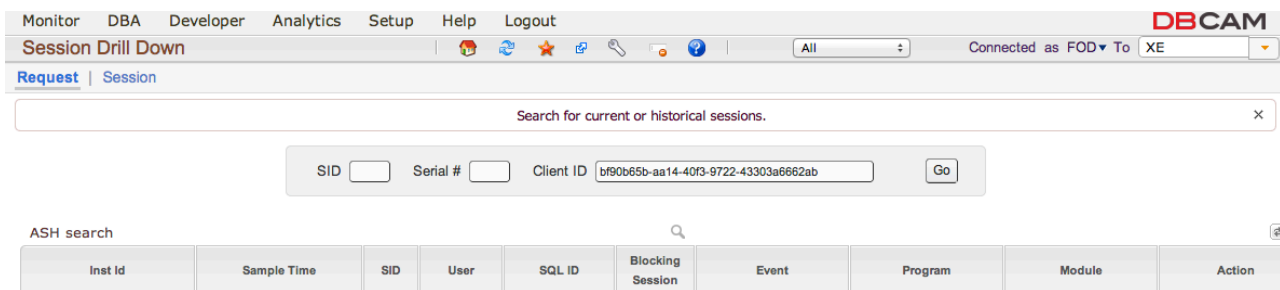


To Link to DB CAM from a Transaction Snapshot

1. In the flow map of a transaction snapshot, right-click on the link below the database icon.
2. Click **Search in DBCam**.



DB CAM launches and displays the session drill down for the selected database.



Learn More

- [Access the Administration Console](#)
- [Dashboards](#)
- [Flow Maps](#)
- [Transaction Snapshots](#)
- [Monitor Databases](#)
- [DB CAM](#)

Integrate AppDynamics with Splunk

- [Prerequisites for Configuring Splunk Integration](#)
 - [To Configure AppDynamics to Interface with Splunk](#)
- [Launching a Splunk Search from AppDynamics](#)
 - [Prerequisites](#)
 - [To launch a Splunk search](#)
- [Seeing Performance Data from AppDynamics in Splunk](#)
- [Viewing AppDynamics Notifications in Splunk](#)
- [Learn More](#)

The AppDynamics-Splunk integration enables you to get a 360-degree view of your application performance by leveraging the AppDynamics "inside-the-app" view and the Splunk "outside-the-app" view.

The integration provides three capabilities. You can do the following:

- Launch Splunk searches using auto-populated queries from in-context locations in the

- AppDynamics Console based on criteria such as time ranges and node IP address.
- Push notifications on policy violations and events from AppDynamics to Splunk so that a Splunk user can use those to launch deep dives in AppDynamics. See [Viewing AppDynamics Notifications in Splunk](#). This enables you to see what was going on in the logs around the time of a specific event as well as navigate to AppDynamics-provided details at that point in time.
- Mine performance data from AppDynamics using the Controller REST API, and push it into Splunk. See [Seeing Performance Data from AppDynamics in Splunk](#).

Watch this webinar:<http://info.appdynamics.com/AppDynamicsSplunkWebinarRecording.html> to hear John Martin, Sr. Director of Production Engineering at Edmunds.com, give an overview of how he uses AppDynamics and Splunk together, to gain the visibility he needs to manage the performance of his award-winning consumer car website, Edmunds.com. This video covers the use cases initiated from Splunk using the AppDynamics App for Splunk.

This topic covers the Splunk search capability that is launched from AppDynamics.

Prerequisites for Configuring Splunk Integration

To use the **Search Splunk** capability, you need to configure AppDynamics to communicate with your Splunk Server. To do this you need the following:

- Proper AppDynamics administrator credentials
 - For single tenant installations, login as the AppDynamics Administrator (root administrator)
 - For multi-tenant installations login as the Account Administrator for the specific tenant using Splunk.
- The URL and port number for the Splunk server.

See [the Splunk website](#) for information about Splunk.

To Configure AppDynamics to Interface with Splunk

- In the upper right corner of the menu bar, click **Settings -> Administration**.
- Click the **Integration** tab.
- Click Splunk in the Integrations list.

The screenshot shows the AppDynamics Administration console. The top navigation bar includes 'Administration' and 'Integrations'. The 'Integrations' tab is selected, showing a list of integrations on the left and a configuration panel for 'Splunk' on the right.

Integrations List:

- splunk> Splunk

Splunk Configuration Panel:

- Enabled:** ☒
- URL:**
- Full URL and port to Splunk server. For example:** `http://mySplunk.server.com:8080/`
- Extra Query Parameters:**
- These query parameters will be appended to the end of any outbound link to Splunk. For example:** `myProp=myValue`
- Save** button

5. Check the **Enabled** check box to enable the integration.
6. Enter the Splunk URL and port number.
7. (Optional) Enter Extra Query Parameters. These parameters are appended to each Splunk search initiated from AppDynamics.
8. Click **Save**.

Launching a Splunk Search from AppDynamics

You can launch a search of your Splunk logs for a specific time frame associated with a transaction snapshot from several points in AppDynamics.

Prerequisites

- Splunk credentials: The first time that you launch a Splunk search, you are prompted to provide your Splunk credentials. After this, your credentials are cached by the browser.
- Your Splunk Server is up and running.
- Your browser is configured to allow pop-ups.

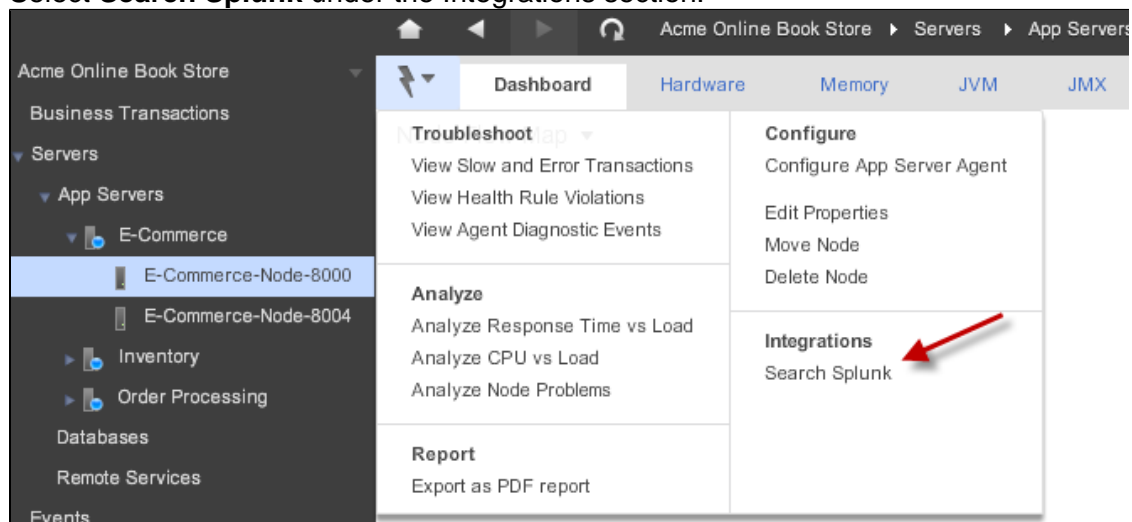
Enable Pop-ups

The first time that you access the Splunk Server, you are prompted to log in. If nothing happens, it is most likely that either your browser is blocking the Splunk login pop-up or the Splunk Server is not running.

To launch a Splunk search

You can access the **Search Splunk** option from the following locations:

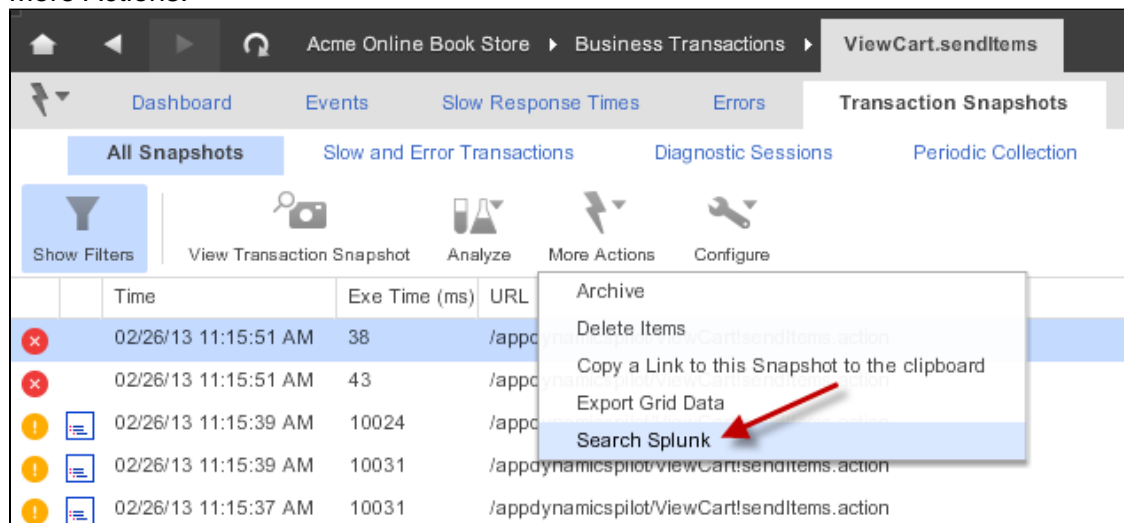
1. The node dashboard:
 - a. Navigate to a node dashboard.
 - b. Click **More Actions**.
 - c. Select **Search Splunk** under the Integrations section.



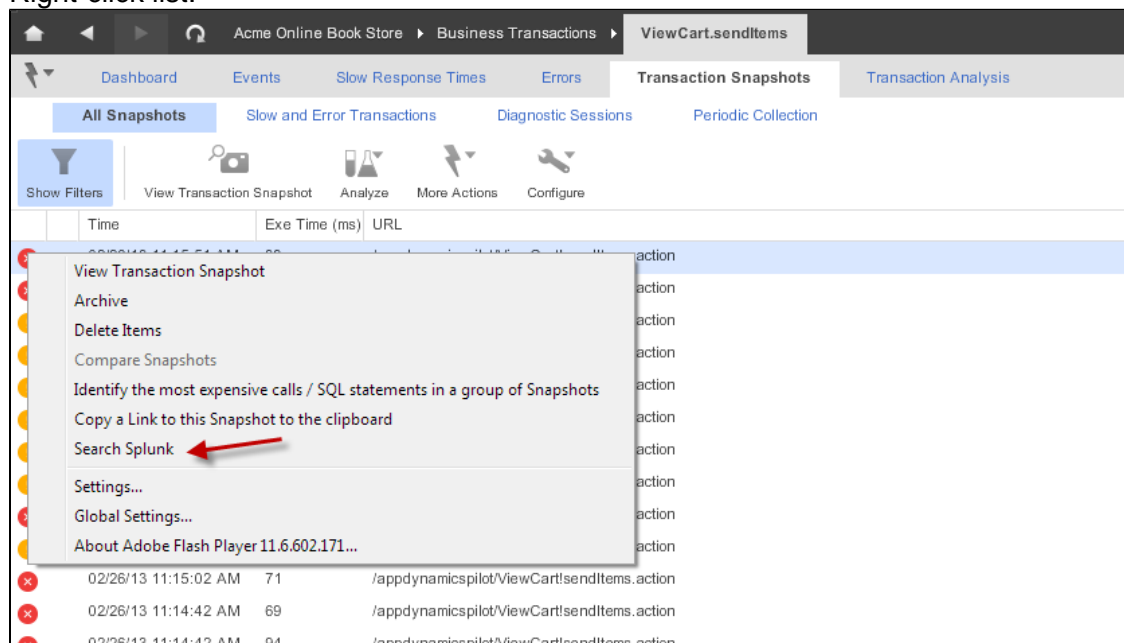
2. The list of transaction snapshots:

- a. From the business transaction dashboard, select the Transaction Snapshot tab.
- b. Select a transaction snapshot and right-click to access a list of options or click **More Actions** to see **Search Splunk** in the list of options.

More Actions:



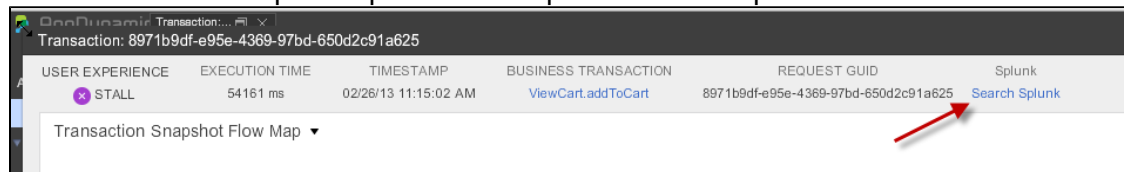
- c. Right-click list:



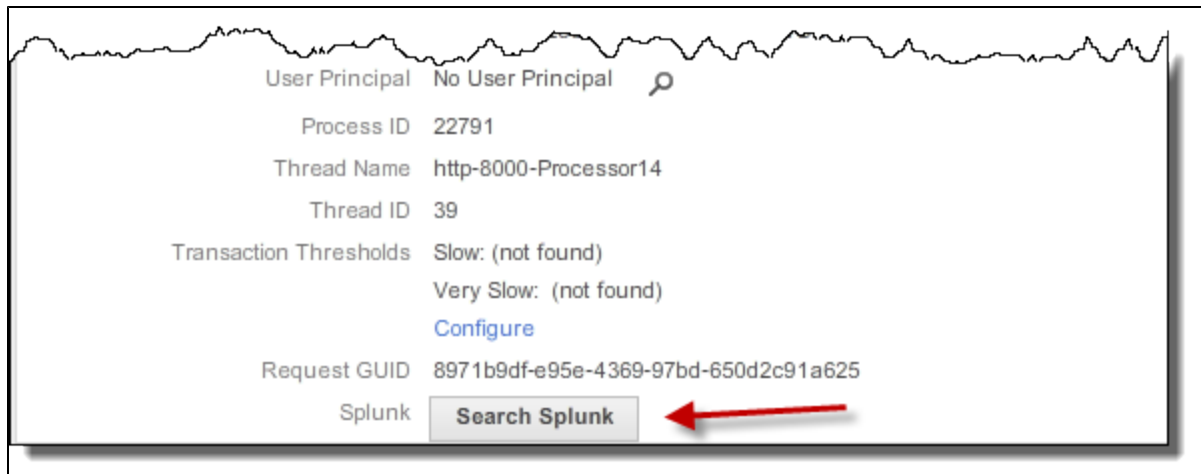
- d. Select **Search Splunk**

3. The Transaction Snapshot Flow Map:

- a. Locate the Search Splunk option at the top of the flow map.



4. The Call Drill Down Summary pane:



Seeing Performance Data from AppDynamics in Splunk

This feature is initiated using [the Splunk eXtension](#).

Viewing AppDynamics Notifications in Splunk

This feature is initiated using [the Splunk eXtension](#).

Learn More

- [Splunkbase Apps](#)
- [Splunk documentation](#)

Configure Custom Metrics for the z-OS Machine Agent

- [Start the Resource Management Facility \(RMF\)](#)
 - [To initialize and start the RMF](#)
 - [To install the scripts](#)
 - [To confirm that the scripts are successful](#)
- [Learn More](#)

This topic describes how to configure custom metrics in a z-OS environment.

Start the Resource Management Facility (RMF)

AppDynamics requires the RMF (Resource Management Facility) to collect the data for the required metrics.

To initialize and start the RMF

1. Connect to the EPTDFRH user and initialize the ETPGZCK user, if not already done.
2. Connect to TSO and provide the details of IBMUSER.
3. Upon a successful connection, choose the **SD (System Display and Search Facility)** option.
4. Use the following commands to initialize and start the RMF:

```
/S RMF
```

```
/F RMF,START III
```

```
/S GPMSERVE,MEMBER=01
```

5. Access the following URL to confirm the RMF startup:

```
http://192.86.32.72:8803/
```

After successful RMF startup, the URL should display a valid HTML page.

6. Click **Explore** and then **Metrics** to display all the available metrics for SVSCPLEX, SYSPLEX.

To install the scripts

1. Download and unzip the attached [zos-machine-agent.zip](#) file to the <machine-agent-install-directory>/monitors/ directory.
2. Select the required metric locations, and add them to urls.list file in the zos-monitor directory.
For example:

```
% CPU utilization          =  
http://192.86.32.72:8803/gpm/perform.xml?resource=S0W1,* ,PROCESSOR&i  
d=8D0460  
% users                    =  
http://192.86.32.72:8803/gpm/perform.xml?resource=%22,SVSCPLEX,SYSPL  
EX%22&id=8D0D50  
% using for i/o by MVS image    =  
http://192.86.32.72:8803/gpm/perform.xml?resource=%22,SVSCPLEX,SYSPL  
EX%22&id=8D1DA0  
% CSA utilization by MVS image  =  
http://192.86.32.72:8803/gpm/perform.xml?resource=%22,SVSCPLEX,SYSPL  
EX%22&id=8D2410  
% users by MVS image          =  
http://192.86.32.72:8803/gpm/perform.xml?resource=%22,SVSCPLEX,SYSPL  
EX%22&id=8D0D60
```

To confirm that the scripts are successful

Once the above steps are performed successfully, start the Machine Agent in debug mode. The following information in the Machine Agent log confirms that the processing is successful.

```
[Worker-7] 28 Mar 2012 19:59:02,128 INFO ExecTask - Started Executable Command
[[G:\AppDynamics\64bit\3.3.4\
MachineAgent-3.3.4.0RC\monitors\CustomMonitor\metrics.bat]]
[Worker-7] 28 Mar 2012 19:59:02,128 DEBUG ExecTask - Will wait for process exit
, before sending execution status.
[Worker-1] 28 Mar 2012 19:59:04,651 DEBUG MonitorOutputHandler - Monitor line
parsed:name=Custom Metrics|zos|% CPU utilization (CP), value=3
[Worker-1] 28 Mar 2012 19:59:04,653 DEBUG MonitorOutputHandler - Reporting
metric after reading metric [Custom Metrics|zos|% CPU utilization (CP)]
[Worker-1] 28 Mar 2012 19:59:04,653 DEBUG MonitorOutputHandler - Reporting
Metric Name [Custom Metrics|zos|% CPU utilization (CP)] Value [3]
[Worker-7] 28 Mar 2012 19:59:04,654 DEBUG ExecTask - Process exited with code: 0
```

Learn More

- [App Agent for Java on z-OS or Mainframe Environments Configuration](#)

Connector Development Guide

- Metadata
 - compute-center-types XML Schema
 - Element Definitions
 - compute-center-type
 - Connector-impl-class-name
 - Property-definition
 - Machine-descriptor-definition
 - image-repository-types
 - Image-types
- Implementing the IConnector Interface
 - void setControllerServices(IControllerServices controllerServices)
 - int getAgentPort()
 - createMachine(IComputer computeCenter, IImage image, IMachineDescriptor machineDescriptor)
 - refreshMachineState(IMachine)
 - restartMachine(IMachine)
 - terminateMachine(IMachine)
 - void deleteImage(IImage image)
 - void refreshImageState(IImage image)
 - void validate(IComputeCenter computeCenter)
 - void configure(IComputeCenter computeCenter)
 - void unconfigure(IComputeCenter computeCenter)
 - void validate(IImageStore imageStore)
 - void configure(IImageStore imageStore)
 - void unconfigure(IImageStore imageStore)
 - void validate(IImage image)
 - void configure(IImage image)

- `void unconfigure(Image image)`
- **Deploying a Connector**

The AppDynamics IConnector interface allows custom implementations of orchestration functionalities such as creating, destroying, restarting, configuring, and validating machine and image instances directly from the AppDynamics Controller user interface.

To create an AppDynamics Connector, the user must implement the IConnector interface and define three sets of xml metadata. The AppDynamics Controller has the ability to dynamically register any new connector every time it starts.

In addition to this topic see the [Cloud Connector API Javadoc](#).

Metadata

The Metadata provides the Controller with the necessary information to register the new compute center, image, and image repository. The Controller uses the metadata to dynamically form the user interface and link the IConnector implementation to the Controller. The three XML files must be named as the following compute-center-types, image-repository-types, image-types.

compute-center-types XML Schema

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="compute-center-types">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="compute-center-type" maxOccurs="unbounded"
minOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="name"/>
              <xs:element type="xs:string" name="description"/>
              <xs:element type="xs:string" name="connector-impl-class-name"/>
              <xs:element name="property-definitions">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="property-definition" maxOccurs="unbounded"
minOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element type="xs:string" name="name"/>
                          <xs:element type="xs:string" name="description"/>
                          <xs:element type="xs:string" name="required"/>
                          <xs:element type="xs:string" name="type"/>
                          <xs:element type="xs:string"
name="default-string-value"/>
                          <xs:element type="xs:string"
name="string-max-length"/>
                          <xs:element type="xs:string"
name="allowed-string-values"/>
                          <xs:element type="xs:string"
name="default-file-value"/>

```

```

        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="machine-descriptor-definitions">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="property-definition" maxOccurs="unbounded"
minOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element type="xs:string" name="name" />
                        <xs:element type="xs:string" name="description" />
                        <xs:element type="xs:string" name="required" />
                        <xs:element type="xs:string" name="type" />
                        <xs:element type="xs:string"
name="default-string-value" />
                        <xs:element type="xs:string"
name="string-max-length" />
                        <xs:element type="xs:string"
name="allowed-string-values" />
                        <xs:element type="xs:string"
name="default-file-value" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

```



```
</xs:element>
</xs:schema>
```

Element Definitions

compute-center-type

List of compute-center-type associated with the IConnector implementation. Each compute-center-type will be registered under the Compute Clouds tab in the UI, for example:

Type	Amazon Elastic Computing Cloud ▼
AWS Account ID *	<input type="text"/>
Access Key *	<input type="text"/>
Secret Access Key *	<input type="text"/>

- **Name:** Compute Center Type name.
- **Description:** Compute Center Type name shown on the Controller GUI.

For example:

```
<description>Amazon Elastic Computing Cloud</description>
```

Connector-impl-class-name

Full name of the IConnector implementation class.

Property-definition

List of property definitions for the compute center. These definitions are used to dynamically generate the property text fields such as AWS Account ID, Access Key, and Secret Access Key.

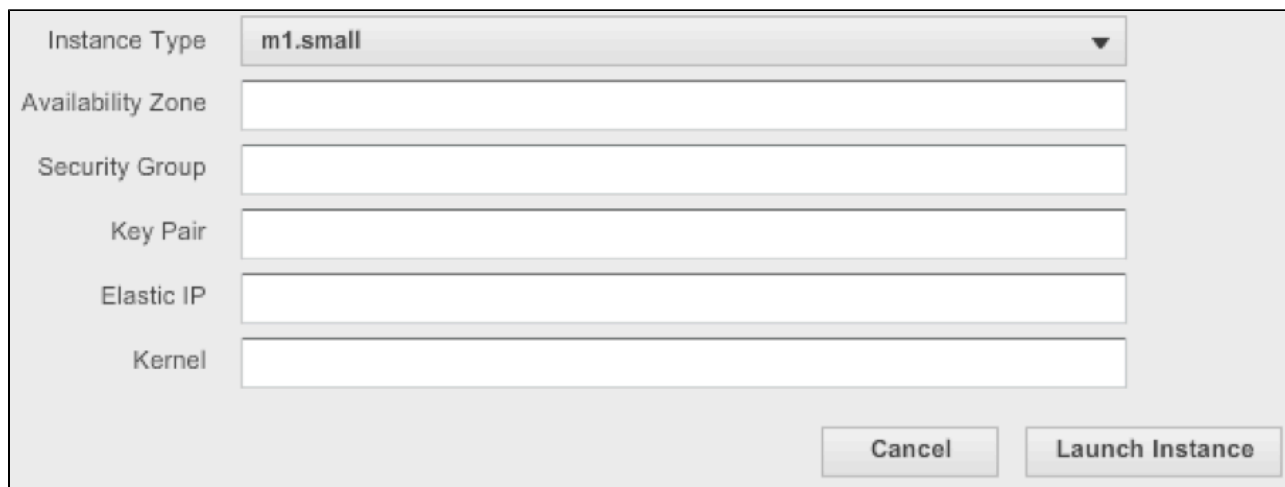
- **Name:** Name of the property field. Such as “AWS Account ID” in Figure 1.
- **Description:** Description of the property.
- **Required:** Checks if the property field cannot be empty. State true/false.
- **Type:** Type of the property field. STRING/FILE
- **Default-string-value:** Default initialization value of the STRING property.
- **String-max-length:** Maximum number of characters allowed to be stored in the property field
- **Allowed-string-values:** List of allowed string values, delimited by comma. If specified this property will be displayed as a drop down list of all the allowed string values.
- **Default-file-value:** Default FILE type property value

For example, using the “AWS Account” property:

```
<property-definition>
  <name>AWS Account ID</name>
  <description>AWS Account ID</description>
  <required>true</required>
  <type>STRING</type>
  <default-string-value></default-string-value>
  <string-max-length>80</string-max-length>
  <allowed-string-values></allowed-string-values>
  <default-file-value></default-file-value>
</property-definition>
```

Machine-descriptor-definition

Contains list of property definitions associated with a machine instance object of that Compute Center type. These properties can be seen in Launch Instance window under the Images tab. For example, an EC2 Launch Instance Window showing Amazon Elastic Computing Cloud Compute Type MachineDescriptor Definitions:



The screenshot shows a 'Launch Instance' window with the following fields and controls:

- Instance Type:** A dropdown menu currently showing 'm1.small'.
- Availability Zone:** An empty text input field.
- Security Group:** An empty text input field.
- Key Pair:** An empty text input field.
- Elastic IP:** An empty text input field.
- Kernel:** An empty text input field.
- Buttons:** 'Cancel' and 'Launch Instance' buttons are located at the bottom right of the window.

image-repository-types

When each compute cloud is registered, a corresponding imagestore object is created. The imagestore name, description, connector-impl-class-name, and property-definitions should mirror the Compute Center property-definitions.

The image-repository-types XML schema:

```

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="image-repository-types">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="image-repository-type" maxOccurs="unbounded"
minOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="name"/>
              <xs:element type="xs:string" name="description"/>
              <xs:element type="xs:string" name="connector-impl-class-name"/>
              <xs:element name="property-definitions">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="property-definition" maxOccurs="unbounded"
minOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element type="xs:string" name="name"/>
                          <xs:element type="xs:string" name="description"/>
                          <xs:element type="xs:string" name="required"/>
                          <xs:element type="xs:string" name="type"/>
                          <xs:element type="xs:string"
name="default-string-value"/>
                          <xs:element type="xs:byte" name="string-max-length"/>
                          <xs:element type="xs:string"
name="allowed-string-values"/>
                          <xs:element type="xs:string"
name="default-file-value"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Image-types

Each image type defines an image type found under the Images tab. For example:

Format	AMI ▼
AMI ID *	<input type="text"/>
Region *	us-east-1 ▼

The image-types XML schema:

```

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="image-types">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="image-type" maxOccurs="unbounded" minOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="name"/>
              <xs:element type="xs:string" name="description"/>
              <xs:element name="property-definitions">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="property-definition">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element type="xs:string" name="name"/>
                          <xs:element type="xs:string" name="description"/>
                          <xs:element type="xs:string" name="required"/>
                          <xs:element type="xs:string" name="type"/>
                          <xs:element type="xs:string"
name="default-string-value"/>
                          <xs:element type="xs:byte" name="string-max-length"/>
                          <xs:element type="xs:string"
name="allowed-string-values"/>
                          <xs:element type="xs:string"
name="default-file-value"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            <xs:element name="supported-compute-center-types">
              <xs:complexType>
                <xs:sequence>
                  <xs:element type="xs:string" name="compute-center-type"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Implementing the IConnector Interface

void setControllerServices(IControllerServices controllerServices)

When the connector implementation is created it is passed a handle to the Controller services. The implementor must maintain a reference to this object if it is needed when invoking other connector operations. This method will be most likely implemented as the following:

```
private IControllerServices controllerServices;  
public void setControllerServices(IControllerServices controllerServices)  
{  
    this.controllerServices = controllerServices;  
}
```

int getAgentPort()

Retrieve the agent port for machine instances managed through this connector. Most likely it is the value returned by calling the controllerServices.getDefaultAgentPort() method. For example:

```
controllerServices.getDefaultAgentPort();
```

createMachine(IComputer computeCenter, IImage image, IMachineDescriptor machineDescriptor)

Creates a new machine instance in the specified compute center, using the specified image. The type of machine instance and the hardware configuration etc, are specified in the machine instance descriptor. The implementation should try to not block the call until the machine instance is fully started. If possible, it should start the machine instance creation and return a Machine handle with the state set to MachineState.STARTING. The refreshMachineState(IMachine) method can then be called to check when machine instance startup is complete. If the Machine handle is created through the IControllerServices interface then its state will automatically be set to MachineState.STARTING. For example:

```

Public IMachine createMachine(IComputeCenter computeCenter, IImage image,
IMachineDescriptor machineDescriptor) throws InvalidObjectException,
ConnectorException
{
//retrieve property definitions stored in IComputeCenter, IImage, and
IMachineDescriptor //objects
String accountId =
controllerServices.getStringPropertyValueByName(computeCenter.getProperties(),
"Account ID");
String accessKey =
controllerServices.getStringPropertyValueByName(computeCenter.getProperties(),
"Access Key");
String imageId =
controllerServices.getStringPropertyValueByName(image.getProperties(), "Image
ID");
String machineSize =
controllerServices.getStringPropertyValueByName(machineDescriptor.getProperties(
), "Image Size");
//To grab a file property, the code must iterate through the corresponding
property //array. For example:
Byte[] bytes = null;
For(IProperty i: image.getProperties)
{
    If (i.getDefinition.getType == PropertyType.FILE)
    {
        If (i.getDefinition.getName().equals("Property Name");
        {
            bytes = ((IFileProperty)i).getFileBytes();
            Break;
        }
    }
}

/*
Code to create a new machine instance on the specified compute center
*/
int agentPort = controllerServices.getDefaultAgentPort();
IMachine machine = controllerServices.createMachineInstance("machine instance
id", "unique internal name", computeCenter, machineDescriptor, image,
agentPort);
return machine;
}

```

refreshMachineState(IMachine)

The refreshMachineState method is used by the controller to poll the status of a particular IMachine object. If the MachineState of an IMachine object is not marked as STARTED, the controller will poll its status at an interval. If the MachineState is set as STARTED, the controller will assume the machine instance is running and will not invoke this method. For example:

```
public void refreshMachineState(IMachine machine) throws InvalidObjectException,
ConnectorException
{

    //get the IMachine instance name, usually stores the corresponding machine
    instance id
    String machineId = machine.getName();

    //Each IMachine object has access to its IComputeCenter, IMachineDescriptor, and
    IImage //objects
    String zone =
    controllerServices.getStringPropertyValueByName(machine.getMachineDescriptor().g
etProperties(), "Zone");

    /*
    Code to grab machine instance status
    Invoke any post build actions
    */
    /*
    The IMachine object Machine state must be set accordingly.

    MachineState.STOPPED will mark the IMachine as stopped and delete the IMachine
    object from the controller. Machinestate can be set manually to STOPPED if the
    machine instance no longer exists
    MachineState.STARTED will set the IMachine as started. The controller will no
    longer poll the machine instance status in STARTED state.
    */

    machine.setState(MachineState.STARTING);
}
```

restartMachine(IMachine)

Restarts the specified machine instance. The implementation should try to not block the call until the machine instance is fully restarted. If possible it should begin the machine restart and return immediately. Afterwards the Machine handle state will be set automatically to MachineState.STARTING. The refreshMachineState(IMachine) method can then be called to check when machine instance restart is complete.

terminateMachine(IMachine)

Terminates the specified machine instance. The implementation should try to not block the call until the machine instance is fully terminated. If possible, it should start the machine termination and return immediately. Afterwards the Machine handle state will be set automatically to MachineState.STOPPING. The refreshMachineState(IMachine) method can then be called to check when machine instance termination is complete.

void deleteImage(IImage image)

Delete the image from the underlying dynamic capacity provider.

void refreshImageState(Image image)

Refreshes the image state of the specified image. This callback is needed since the image save and image copy can take a considerable amount of time to complete. Instead of blocking on the image save and image copy operations, the connector implementation should return quickly. Afterwards, the controller will poll the image state through this operation to see when the image save or image copy is complete. When complete, it is the responsibility of the connector implementation to set the image state to ImageState.READY.

void validate(IComputeCenter computeCenter)

Validates if a Compute Center has the valid Properties and rest of the fields. For example:

```
public void validate(IComputeCenter computeCenter) throws
InvalidObjectException, ConnectorException
{

    String accountId =
    controllerServices.getStringPropertyValueByName(computeCenter.getProperties(),
    "AWS Account ID");

    String accessKey = controllerServices.getStringPropertyValueByName(computeCenter
    .getProperties(), "Access Key");

    String secretKey = controllerServices.getStringPropertyValueByName(computeCenter
    .getProperties(), "Secret Access Key");

    /*
    Validate if the user credentials are correct. Throw exception if properties are
    invalid.
    */

}
```

void configure(IComputeCenter computeCenter)

Does any configuration that needs to be done for the Compute Center to be able to be used by Singularity Dynamic Provisioning Manager. This may include creating access accounts, creating permissions, etc.

void unconfigure(IComputeCenter computeCenter)

Un-does any configuration that was done for the Compute Center to be able to be used by Singularity Dynamic Provisioning Manager. This may include deleting access accounts, deleting permissions, etc.

void validate(ImageStore imageStore)

Validates if an Image Store has the valid Properties and rest of the fields.

void configure(IImageStore imageStore)

Does any configuration that needs to be done for the Image Store to be able to be used by Singularity Dynamic Provisioning Manager. This may include creating access accounts, creating permissions, etc.

void unconfigure(IImageStore imageStore)

Un-does any configuration that was done for the Image Store to be able to be used by Singularity Dynamic Provisioning Manager. This may include deleting access accounts, deleting permissions, etc.

void validate(IImage image)

Validates if an Image has the valid Properties and rest of the fields.

void configure(IImage image)

Does any configuration that needs to be done for the Image to be able to be used by Singularity Dynamic Provisioning Manager. This may include creating access accounts, creating permissions, etc.

void unconfigure(IImage image)

Un-does any configuration that was done for the Image to be able to be used by Singularity Dynamic Provisioning Manager. This may include deleting access accounts, deleting permissions, etc.

For more information regarding the IController interface and its related classes, see the [Cloud Connector API Javadoc](#).

Deploying a Connector

To deploy a connector

1. Compile the IConnector implementation into a Jar file.
2. Package the compiled jar and the xml metadata into one directory.
3. Put the directory in: <controller-install-dir>/lib/connectors.
4. Restart the Controller by running the stopController and startController scripts under <controller-install-dir>/bin.

You should see the new connector in the Compute Clouds and Images tab under Systems.