

# AppDynamics Pro Documentation

## Configure AppDynamics



1. Hierarchical Configuration Model .....	4
2. Configure Remote Monitoring of an On-Premise Controller .....	5
3. Configure Error Detection .....	8
4. Configure Baselines .....	15
5. Configure Business Transaction Detection .....	17
5.1 Match Rule Conditions .....	24
5.2 Regular Expressions In Match Conditions .....	25
5.3 Messaging Entry Points .....	25
5.4 Web Entry Points .....	27
5.5 Creating New Tiers .....	31
6. Configure Transaction Snapshots .....	32
7. Configure Data Collectors .....	32
8. Configure Thresholds .....	37
9. Configure End User Experience .....	39
9.1 Configure the .NET App Agent for EUM .....	41
9.2 Configure Page Identification and Naming .....	42
9.3 Configure JavaScript and Ajax Error Detection .....	44
9.4 Configure EUM Thresholds .....	48
9.5 Configure Browser Snapshot Collection .....	48
9.6 Customize Your EUM Deployment .....	49
9.6.1 Add Information to a Browser Snapshot .....	52
9.6.2 Configure Page Names of Any Format .....	53
9.6.3 Handle the window.onerror Event .....	53
9.7 Inject the JavaScript Agent for EUM .....	54
9.7.1 Injection Overview .....	54
9.7.2 Manual Injection .....	57

9.7.3 Automatic Injection	58
9.7.4 Assisted Injection-Using Injection Rules (Java Only)	61
9.7.5 Assisted Injection-Using Attribute Injection	63
9.7.6 Selecting a JavaScript for EUM Instrumentation Strategy	66
10. Configure Background Tasks	67
11. Configure Backend Detection	69
11.1 Configure Custom Exit Points	74
12. Configure Stale Backend Removal	79
13. Configure Policies	80
14. Configure Health Rules	83
15. Configure Email Digests	88
16. Configure Call Graphs	91
17. Configure Business Metric Information Points	93
18. Configure Code Metric Information Points	96
19. Configure AppDynamics for Java	96
19.1 Configure Custom Exit Points (Java)	97
19.1.1 Configurations for Custom Exit Points	101
19.2 Code Metric Information Points (Java)	104
19.3 Configure JMX Metrics from MBeans	105
19.3.1 Exclude JMX Metrics	111
19.3.2 Exclude MBean Attributes	111
19.3.3 Configure JMX Without Transaction Monitoring	112
19.3.4 Resolve JMX Configuration Issues	112
19.3.5 Import or Export JMX Metric Configurations	117
19.4 Configure Custom Memory Structures (Java)	120
19.5 Configure Object Instance Tracking (Java)	124
19.6 Configure Memory Monitoring (Java)	126
19.7 Configure Multi-Threaded Transactions (Java)	127
19.8 Configure Background Tasks (Java)	128
19.9 Web Application Entry Points	130
19.9.1 Servlet Entry Points	131
19.9.1.1 Servlet Transaction Detection Scenarios	140
19.9.1.1.1 Identify Transactions Based on DOM Parsing Incoming XML Payload	140
19.9.1.1.2 Identify Transactions Based on POJO Method Invoked By a Servlet	142
19.9.1.1.3 Identify Transactions for Java XML Binding Frameworks	144
19.9.1.1.4 Identify Transactions Based on JSON Payload	146
19.9.2 Struts Entry Points	148
19.9.3 Web Service Entry Points	149
19.9.4 POJO Entry Points	151
19.9.5 Spring Bean Entry Points	156
19.9.6 EJB Entry Points	158
19.9.7 POCO Entry Points	160
19.9.10 Getter Chains in Java Configurations	164
20. Java Server-Specific Installation Settings	166
20.1 Apache Cassandra Startup Settings	167
20.2 Apache Tomcat Startup Settings	168
20.2.1 Tomcat as a Windows Service Configuration	172
20.3 Glassfish Startup Settings	173
20.4 IBM WebSphere Startup Settings	175
20.4.1 App Agent for Java on z-OS or Mainframe Environments Configuration	178
20.5 JBoss Startup Settings	181
20.6 Jetty Startup Settings	187
20.7 Oracle WebLogic Startup Settings	187
20.8 OSGi Infrastructure Configuration	191
20.9 Resin Startup Settings	193
20.10 Solr Startup Settings	195
20.11 Standalone JVM Startup Settings	196
20.12 Tanuki Service Wrapper Configuration	197
20.13 Tibco BusinessWorks Configuration	197
21. App Agent for Java Configuration Properties	198
22. Configure App Agent for Java for Batch Processes	205
23. Configure App Agent for Java for JVMs that are Dynamically Identified	206
24. Configure App Agent for Java in Restricted Environments	207
25. Configure App Agent for Java in z-OS or Mainframe Environments	207
26. Configure App Agent for Java on Multiple JVMs on the Same Machine that Serve Different Tiers	210
27. Configure App Agent for Java on Multiple JVMs on the Same Machine that Serves the Same Tier	211
28. Configure App Agent for Java to Use Existing System Properties	213
29. Configure AppDynamics for .NET	215
29.1 Configure Custom Exit Points (.NET)	216
29.2 Getter Chains in .NET Configurations	221
29.3 Enable Thread Correlation (.NET)	221
29.4 Configure the .NET Machine Agent	222
29.5 Enable Correlation for .NET Remoting	224
30. Configure the App Agent for .NET	225
31. How to Configure the .NET Agent Manually	239

32. App Agent for .NET Configuration Properties .....	242
33. Configure Authentication, User Permissions and Integrations .....	248
33.1 Configure Authentication Provider .....	249
33.2 Configure Authentication Using SAML .....	249
33.2.1 Disable SAML Authentication for an Account .....	251
33.2.2 SAML Configuration for OneLogin .....	252
33.3 Configure Authentication Using LDAP .....	255
33.4 Configure Users .....	260
33.5 Configure Groups .....	262
33.6 Configure Roles .....	263
33.6.1 Access Role Configuration .....	265
33.6.2 View Predefined Roles .....	265
33.6.3 Configure Custom Roles .....	265
33.6.4 Assign Roles to Users and Groups .....	269
33.7 Configure Integrations .....	269
34. Export and Import Business Application Configurations .....	270
35. Configure File Descriptor Limits on Linux .....	271
36. Configure Swappiness on Linux .....	272
37. Configure the Machine Agent to Automatically Start on Linux .....	273
38. Configure Custom Metrics for the z-OS Machine Agent .....	274
39. Machine Agent Configuration Properties .....	276
40. Configure Multiple Machine Agents for One Machine .....	281
41. Configure an On-Premise Controller .....	283
41.1 Controller Licenses .....	283
41.2 Controller Tenant Mode .....	283
41.3 Controller Port Settings .....	285
41.4 Controller SSL and Certificates .....	287
41.5 Controller Logs .....	287
41.6 Controller Performance .....	290
41.7 Configure the SMTP Server .....	291
41.8 Controller High Availability .....	292
41.8.1 Manage Controller High Availability .....	292
41.8.2 Common Commands Used to Set Up Controller HA .....	296
41.8.3 Controller HA Failover and Failback Process .....	296
41.8.3.1 Automating HA Controller Failover and Failback .....	299
41.8.4 Controller High Availability FAQ .....	301
41.8.5 Provisioning Controllers in High Availability Mode .....	302
41.8.5.1 Setting Up an SSH Key for Controller Provisioning .....	304
41.9 Controller Data and Backups .....	305
41.9.1 Controller Data Backup and Restore .....	305
41.9.2 Controller Database Scripts .....	308
41.9.3 Controller Data Storage .....	308
41.9.4 Controller Disk Space and the Database .....	309
41.9.5 Database Size, Data Retention, and Metric Resolution .....	310
42. Controller Info Configuration for Agents .....	314
43. Configure Controller VIP .....	315

# Hierarchical Configuration Model

- Entry Point and Exit Point Inheritance
- Node Inheritance
- Switching Configuration Levels
- Learn More

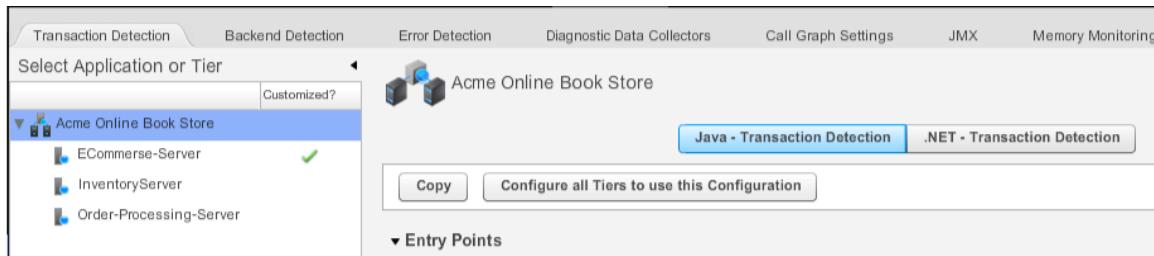
Transaction detection (entry point), backend detection (exit point), and node property configurations are applied on a hierarchical inheritance model. This model provides a default configuration for new tiers as well as the ability to re-use custom configurations in all tiers or tiers that you specify, eliminating the need to configure custom entry and exit points for all tiers.

A tier can inherit all its transaction detection and backend detection configuration from the application, or it can override the application configuration to use a custom configuration.

Similarly, a node can inherit its entire node property configuration from its parent, or it can override the parent configuration to use a custom configuration.

## Entry Point and Exit Point Inheritance

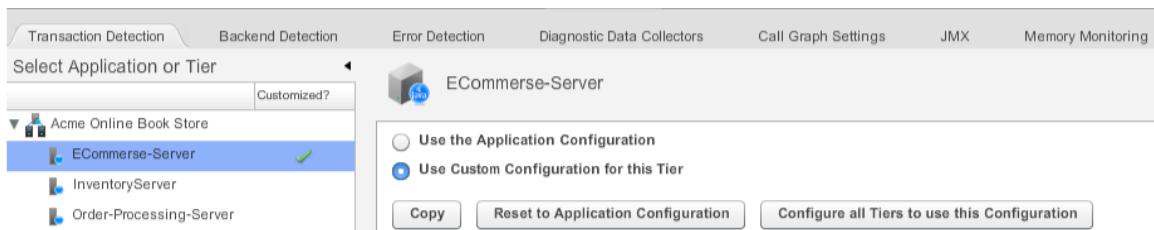
By default, tiers inherit the entry point and exit point configurations of the application. You can copy the application-level configuration to specific tiers or explicitly configure all tiers to use the application-level configuration.



At the tier level, you can specify that the tier should use the application-level configuration.

Or you can override the application-level configuration by creating a custom configuration for the specific tier.

You can configure all tiers to use the custom configuration or copy the configuration for re-use in specific tiers. You can also reset a tier that is currently using a custom configuration to use the application-level configuration.



## Node Inheritance

By default a node inherits the node properties of its parent tier (or of the application).

When you configure node properties you can specify that all nodes in a tier inherit the node properties of the parent (tier or application) or that the node should use a custom configuration.

The screenshot shows the 'App Server Agent Configuration' window. On the left, there's a tree view under 'Select Application, Tier, Node' with 'Customized?' checked. It shows 'CartApp\_131' expanded, with 'ecom', 'inven', and 'order' as children. On the right, there's a section for 'I-node' with two radio buttons: 'Use Parent Configuration (Tier or Application)' (selected) and 'Use Custom Configuration'.

If you create a custom configuration for a node, you can copy that configuration to the application, tier or to another node.

## Switching Configuration Levels

If you customize configuration at the tier or node level and then switch back to the application-level configuration, you will not see the old configuration in the UI. However, the old tier or node level configuration is stored, and if you will see these old settings if you switch to the lower-level configuration again.

## Learn More

- [Configure Backend Detection](#)
- [Configure Business Transaction Detection](#)
- [App Agent Node Properties](#)

# Configure Remote Monitoring of an On-Premise Controller

- [Prerequisites](#)
- [Configuring Remote Monitoring](#)
  - Make a formal request to AppDynamics Support
  - Shut down your Controller and its application server
  - Modify the Controller configuration
  - Create a new controller-info.xml file for the application agent
    - Sample controller-info.xml file
  - Restart the Controller's application server
  - Install the Machine Agent for the Controller
  - Monitor the performance of the Controller
- [Learn More](#)

This topic describes how to configure your Controller so that it can be monitored remotely for optimization purposes.

## Prerequisites

Install the AppDynamics Machine Agent on the machine where your Controller is installed. See [Install the Machine Agent](#).

## Configuring Remote Monitoring

Follow these instructions to configure an on-premise Controller so that you can remotely monitor your AppDynamics environment.

### Make a formal request to AppDynamics Support

To help you optimize your Controller, AppDynamics Support will set up an account for you on our monitoring system.

You will receive the following information from the AppDynamics Support team:

- Account name
- Account key
- Application name



**Important:** Wait for the response from the AppDynamics Support before proceeding.

## Shut down your Controller and its application server

1. Shut down your Controller. Open a command line console and execute following command:

- For Linux:

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

- For Windows:

```
controller.bat stop
```

2. Shut down the Controller's application server. Open a console and navigate to the <Controller\_Install\_Directory>/bin directory and execute following command:

For Linux:

```
./controller.sh stop-appserver
```

For Windows:

```
controller.bat stop-appserver
```

## Modify the Controller configuration

1. Open the <Controller\_Installation\_Directory>/appserver/domains/domain1/conf/domain.xml file, the configuration file for the Controller's application server.

2. Update the values of the following JVM options as shown:

```
-Dappdynamics.controller.hostName=saaS-monitor.saas.appdynamics.com  
-Dappdynamics.controller.port=80
```

If SSL is required add:

```
-Dappdynamics.controller.ssl.enabled=true
```

See [Controller SSL and Certificates](#).

## Create a new controller-info.xml file for the application agent

This step requires the account name and key sent to you by the AppDynamics Support Team.

1. Create a new controller-info.xml file. Refer to the sample controller-info.xml file shown below.  
If SSL is required set controller.ssl.enabled to "true".
2. Add the new controller-info.xml file to the <Controller\_Installation\_Directory>/appserver/domains/domain1/appagent/conf folder.
3. Set the account name to the account name supplied by AppDynamics Support.
4. Set the application name to the application name supplied by AppDynamics Support.

*Optional:* Modify the node name to the name of the machine hosting the Controller.

 **Important:** The tier name must be "App Server". Do not change this value.

### Sample controller-info.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<controller-info>
    <controller-host>saas-monitor.saas.appdynamics.com</controller-host>
    <controller-port>80</controller-port>
        <controller-ssl-enabled>false</controller-ssl-enabled>
    <disable-virtualization-resolvers>true</disable-virtualization-resolvers>
    <account-name>testing</account-name>
        <account-access-key>BlueCust!20</account-access-key>
    <application-name>Controller</application-name>
    <tier-name>App Server</tier-name>
    <node-name>mynode</node-name>
    <agent-install></agent-install>
    <agent-runtime-dir></agent-runtime-dir>
</controller-info>
```

Modify only those values that are highlighted in the sample file given above.

## Restart the Controller's application server

1. Open a console and navigate to the <Controller\_Install\_Directory>/bin directory and execute following command:

- For Linux:

```
./controller.sh start-appserver
```

- For Windows:

```
controller.bat start-appserver
```

## Install the Machine Agent for the Controller

1. Install the Machine Agent on the Controller machine. See [Install the Machine Agent](#).
2. Use the same controller-info.xml file used by the App Agent, to point the Machine Agent at the AppDynamics Monitor.

## Monitor the performance of the Controller

1. Open a browser at <http://saas-monitor.saas.appdynamics.com/controller/>.
2. Use the following information to access the Controller UI:
  - Account: <account-name> (same as specified in the URL)
  - User: admin (You can request an alternative user name to the one supplied by Support in step 1).
  - Password: (supplied by Support in step 1)
3. View the performance of your Controller.

## Learn More

- [Controller SSL and Certificates](#)

## Configure Error Detection

- [Error Identification](#)
  - [Error Detection Using Logged Exceptions or Messages](#)
    - To configure error detection using logs for Java
    - To configure error detection using logs, system trace, and events for .NET
  - [Custom Logger Definitions](#)
    - To define a custom logger
  - [Configuring Exceptions to Ignore](#)
- [Configuring HTTP Response Code Errors](#)
- [Configuring Error Redirect Pages](#)
- [Learn More](#)

You can configure which exception or error codes should be ignored or mapped to a custom name.

## Error Identification

You can configure:

- where AppDynamics detects errors
- exceptions and messages to ignore
- HTTP return codes to define as errors
- error detection using redirect pages
- custom loggers

The error detection configuration is applied to all tiers in the application.

## Error Detection Using Logged Exceptions or Messages

By default AppDynamics instruments Java error and warning methods such as logger.warn and logger.error. AppDynamics captures the exception stack trace and automatically correlates it with the request.

### To configure error detection using logs for Java

For details about the Java logging APIs see [Java Logging](#).

1. Click **Configure -> Instrumentation**.

2. Click the **Java - Error Detection** tab.

The screenshot shows the 'Error Detection Using Logged Exceptions or Messages' section. It includes a heading, a sub-section title, and a list of five checkboxes:

- Detect errors logged using java.util.logging (Java 1.6 is required)
- Detect errors logged using Log4j
- Detect errors by looking at messages logged with ERROR or higher
- Mark Business Transaction as error

3. Select where you want AppDynamics to look for log messages and exceptions.

If you want AppDynamics to count all business transactions with messages logged with ERROR or higher as error transactions, check the Mark Business Transaction as error checkbox. Clear the checkbox if you do not want any business transactions with these messages counted as errors. If you want to exclude only specific messages from causing the business transaction to be counted as an error transaction, check the checkbox and add the messages that you want to exclude in the Ignored Messages list below.

Messages logged as higher than ERROR would include more severe levels such as CRITICAL or FATAL.

4. Click **Save Error Configuration**.

## To configure error detection using logs, system trace, and events for .NET

1. Click **Configure -> Instrumentation**.

2. Click the **.NET - Error Detection** tab.

The screenshot shows the 'Error Detection Using Logged Exceptions or Messages' section. It includes a heading, a sub-section title, and a list of nine checkboxes:

- Detect errors logged using NLog
- Detect errors logged using Log4Net
- Disable System Trace
- Disable Event Log
- Detect errors by looking at messages logged with ERROR or higher
- Mark Business Transaction as error

3. Select where you want AppDynamics to look for log messages and exceptions.
4. If you do not want system trace or event log messages to be monitored, check the appropriate box.
5. If you want AppDynamics to count all business transactions with messages logged with ERROR or higher as error transactions, check the Mark Business Transaction as error checkbox. Clear the checkbox if you do not want any business transactions with these messages counted as errors. If you want to exclude only specific messages from causing the business transaction to be counted as an error transaction, check the checkbox and add the specific messages that you want not to cause error transactions in the Ignored Messages list.
6. Click **Save Error Configuration**.

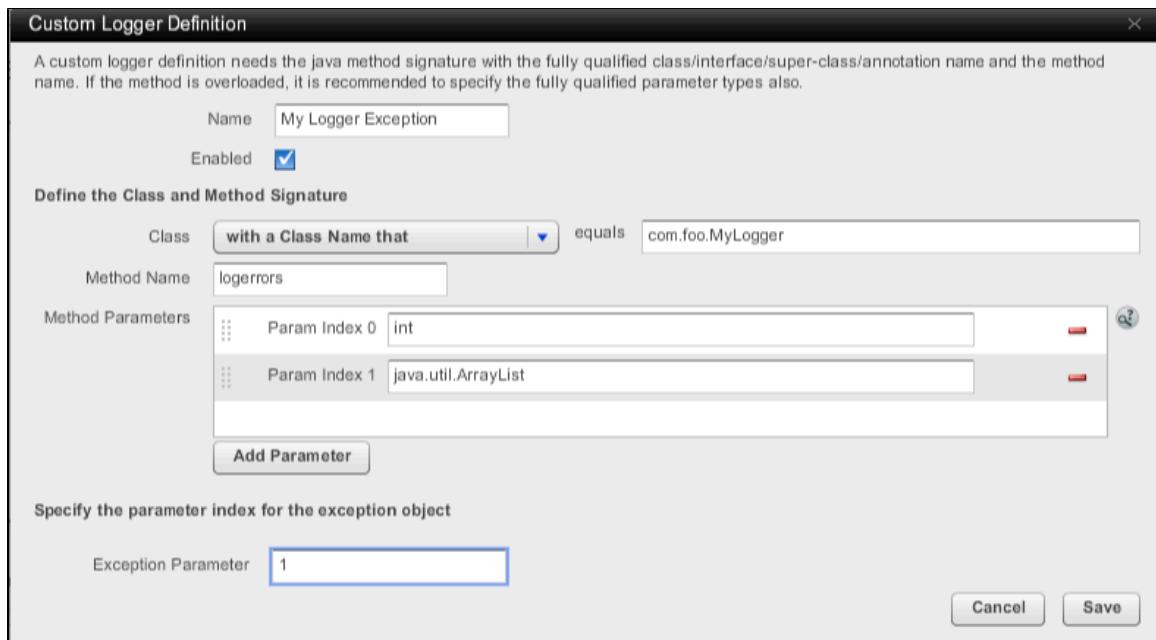
## Custom Logger Definitions

To configure AppDynamics to look for logged errors in a custom logger, create a custom logger definition.

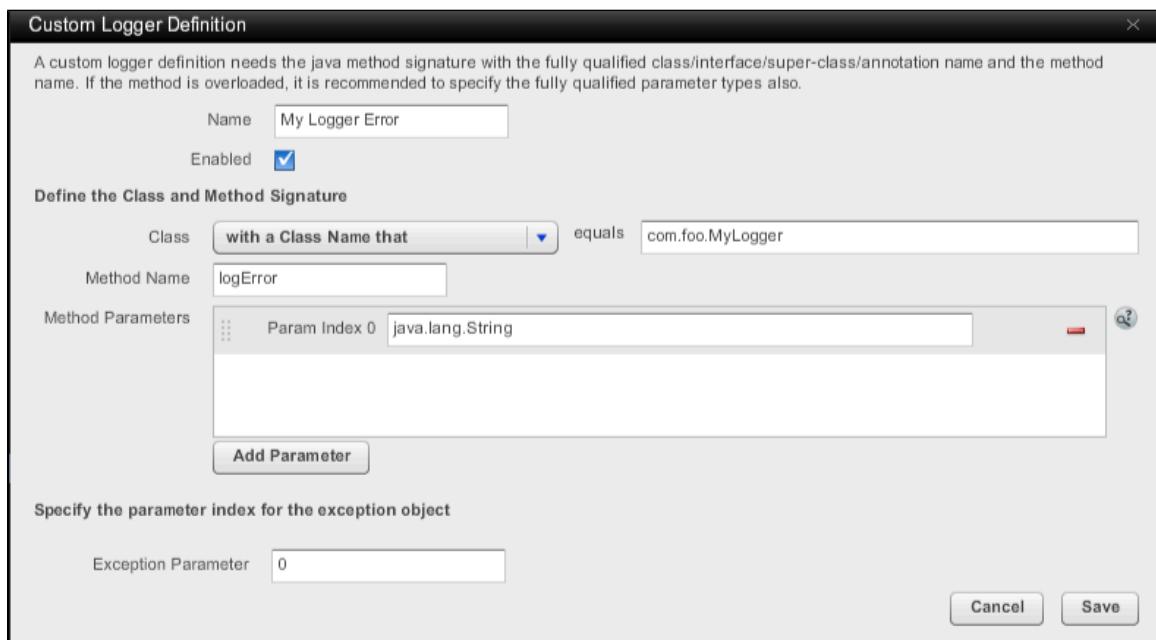
### To define a custom logger

1. In the error configuration screen, click **Add Custom Logger Definition**.  
The Custom Logger Definition window opens.
2. Enter a name for the custom logger.
3. Check the Enabled check box to enable it.
4. Use the drop-down menu and text field to define the logger's class.
5. Enter the name of the method that implements the logger.
6. Click **Add Parameter** to add a parameter to the logger.
7. In the parameter field, enter the fully qualified class name for the parameter.
8. Repeat steps 6 and 7 as necessary to define all of the logger method's parameters.
9. In the Exception Parameter field, enter the zero-based index of the parameter to use to identify the exception. This index can be any type of Object, including Arrays. If the Object is not null, AppDynamics converts the Object to a string to report the error details.
10. Click **Save**.

The following screenshot shows a logger definition for a method with two parameters. It uses the array parameter to describe the exception.



The next screenshot shows a logger definition for a method with one parameter. It uses a string parameter to describe the error details.



## Configuring Exceptions to Ignore

Configuring exceptions and log messages to ignore is useful for excluding a temporary known issue from the error count to avoid skewing error metrics and triggering unnecessary alerts from policies based on the error count. You can configure AppDynamics to ignore specified exceptions and logged messages.

When you configure an exception to be ignored, AppDynamics still detects the exception, logs the exception, increments the exception count, and displays the exception in the **Exceptions** subtab of the **Errors** tab of the tier and node dashboards and in the **Summary** and **Error Details** sections of any transaction snapshot that was in progress when the exception occurred. However, in the transaction snapshot the User Experience item in the transaction Summary would not be marked as ERROR. Ignoring an exception simply means that AppDynamics does not increment the business transaction error count for errors caused by exceptions that have been configured to be ignored. When an ignored exception occurs, AppDynamics does not count the business transaction in which the exception occurred as an error.

▼ Define exceptions or log messages to ignore when detecting error Transactions

**Ignored Exceptions**

Ignore these exceptions when detecting errors

**Add New Exception to Ignore**

**Ignored Messages**

Ignore these logged messages

**Add**

You can specify exception chain, using colons as separators.

Configure an Exception to Ignore when Detecting Errors

**Exception, or Exception Chain**

Enter fully qualified Class Names for Exceptions separated by :

java.net.WebException : com.xyz.ccym.cap.exceptions.CarrierBusinessRulesException: ?

You can direct AppDynamics to ignore an exception only when the exception.getMessage() call contains the string "format", or does not contain the string "format".

Configure an Exception to Ignore when Detecting Errors

**Exception, or Exception Chain**

Enter fully qualified Class Names for Exceptions separated by :

java.net.WebException : com.xyz.ccym.cap.exceptions.CarrierBusinessRulesException: ?

**Match Condition for Exception Message**

This condition will be used to match against exception.getMessage(). If it matches, for the exception configured above, the exception will not be detected as an Error

Exception Message

Equals

format

NOT Condition

Selecting this will reverse the condition and return true if NOT (condition)

The following configuration directs AppDynamics to ignore all java.lang.Runtime exceptions that wrap a javax.sql.SQLException. Other types of java.lang.Runtime exceptions will not be ignored.

**Configure an Exception to Ignore when Detecting Errors**

**Exception, or Exception Chain**

Enter fully qualified Class Names for Exceptions separated by : [?](#)

**Match Condition for Exception Message**

This condition will be used to match against exception.getMessage(). If it matches, for the exception configured above, the exception will not be detected as an Error

Exception Message  |

The next configuration is defined the exceptions to ignore more narrowly. It directs AppDynamics to ignore all java.lang.RuntimeExceptions that wrap a javax.sql.SQLException only when the exception.getMessage() call contains the string "format".

**Configure an Exception to Ignore when Detecting Errors**

**Exception, or Exception Chain**

Enter fully qualified Class Names for Exceptions separated by : [?](#)

**Match Condition for Exception Message**

This condition will be used to match against exception.getMessage(). If it matches, for the exception configured above, the exception will not be detected as an Error

Exception Message  |  format

You can also specify Java logged messages to ignore based on a match condition.

**Ignored Messages**

Ignore these logged messages

You can specify that errors logged with certain loggers or logger categories be ignored.

## Configuring HTTP Response Code Errors

HTTP response codes convey errors that occur while the application is serving user requests as well as business errors.

For example, a 404 error denotes a page not found where as a 430 error might represent an item out of stock situation.

By default, AppDynamics captures all error codes from 400 to 505 and marks them as errors. You need to configure error custom codes ranges and custom error codes with custom descriptions.

The following figure shows the error configuration for ACME Online store. This configuration causes AppDynamics to detect the Inventory Empty Error and Custom Error codes.

Description	Lower Bound (inclusive)	Upper Bound (inclusive)	Enabled
Inventory Empty Error	512	520	<input checked="" type="checkbox"/>
Custom Error Code	522	522	<input checked="" type="checkbox"/>

To suppress an error code if you do not want it to appear, add it to the list by creating a custom error code range and then disable that error code by clearing its Enabled checkbox.

## Configuring Error Redirect Pages

Applications can use custom error pages to redirect to identify application errors.

To specify a redirect page, click **Add Error Redirect Page**.

The screenshot shows a table titled "Detect Errors based on Redirect Pages". The table has four columns: "Name", "Match Criteria", "Match Pattern", and "Enabled". A single row is present in the table:

Name	Match Criteria	Match Pattern	Enabled
CustomErrorHandler	Equals	AcmeErrorHandler.jsp	<input checked="" type="checkbox"/>

Below the table are two buttons: "Add Error Redirect Page" and "Delete".

In this example, whenever an error in the system redirects to "AcmeErrorHandler.jsp", AppDynamics logs an error.

## Learn More

- [Troubleshoot Errors](#)

## Configure Baselines

- [Configuring Baselines](#)
  - To configure an existing baseline pattern
  - To set a baseline as the default baseline
  - To create a new baseline pattern
  - To use data from a specific time as a baseline
- [Learn More](#)

This topic discusses how to configure baselines against which you monitor the performance of your application. For an overview of baselines see [Behavior Learning and Anomaly Detection](#).

## Configuring Baselines

You can define one or more baselines depending on the trends that you want to monitor. You can configure any one of your baselines to be the default baseline for defining health rules.

### To configure an existing baseline pattern

1. Click **Configure-> Baselines**.

AppDynamics lists all available baseline patterns:

The screenshot shows the 'Baselines' configuration screen. At the top, there are buttons for 'New' and 'Delete'. Below is a table with columns: Name, Trend, Time Period, and Default. A row for 'Daily Trend - Last 30 days' is selected, indicated by a blue background and a green checkmark in the Default column.

Name	Trend	Time Period	Default
All data - Last 15 days	None	Last 15 days	
Daily Trend - Last 30 days	Daily	Last 30 days	<input checked="" type="checkbox"/>
Weekly Trend - Last 3 months	Weekly	Last 90 days	
Monthly Trend - Last 1 year	Monthly	Last 365 days	

**Detailed View for Daily Trend - Last 30 days:**

- Name:** Daily Trend - Last 30 days
- Trend:**  Daily (selected),  None - Average data for the whole time period,  Weekly,  Monthly
- Time Period:** Dynamic - Use a rolling time window
  - Use all available data
  - Use the last  days (Min: 1 Max: 31)

At the bottom right are buttons: Delete, Set as Default, and Save.

2. Select the baseline pattern that you want to modify.

AppDynamics displays the configuration details.

3. Select the time period you want to use for the baseline pattern.

4. Select the data that you want to use for baseline calculation. You can also specify the data to be selected from "number of days".

5. Click **Save**.

## To set a baseline as the default baseline

**⚠** The default baseline is used by all existing and future health rules. Be aware of your existing baselines and health rule definitions before you select this option.

1. Click **Configure-> Baselines**.

2. Select the baseline pattern that you want to be the new default baseline.

3. Click **Set as Default**.

4. Click **Save**.

## To create a new baseline pattern

1. Click **Configure-> Baselines**.

2. Click **New**.

3. Enter a **Name**.
4. Select a **Trend**.
5. Select the **Time Period** you want to use for the baseline pattern.
6. Select the data that you want to use for baseline calculation. You can also specify the data to be selected from "number of days".
7. Click **Save**.

## To use data from a specific time as a baseline

1. Click **Configure-> Baselines**.
2. Click **New**.
3. Enter a **Name**.
4. The **Trend** can be either periodic or none.
5. Set the **Time Period** for your baseline as "Fixed - Use a specific time range".
6. Set the date ranges.
7. Click **Save**.

## Learn More

- Behavior Learning and Anomaly Detection
- Configure Health Rules

# Configure Business Transaction Detection

- Planning Business Transaction Detection Configuration
- Basic Configurations
  - To access business transaction detection configuration
- Entry Points
- Exclude Rules
  - Default Exclude Rules
    - To Create Custom Exclude Rules
    - To View Default Exclude Rules
      - Default Exclude Rules for Java Platforms
      - Default Exclude Rules for .NET Platforms
  - Custom Match Rules
    - To Create Custom Match Rules
      - Sample Custom Match Rule
  - Transaction Splitting
    - Sample Custom Match Rule with Split
  - Sequence and Precedence for Auto-Naming and Custom Match Rules
    - The Priority Parameter
  - Transaction Detection Concepts
  - Learn More

## Planning Business Transaction Detection Configuration

Before you configure transaction discovery, select and name the operations you want to monitor. For example, you could:

- Interview the team responsible for each tier to identify the key 5 or 10 or 20 operations that they feel are important to monitor. Identify which key operations must work well for the tier to be a successful part of the site.

- Map the operations that you have identified to the appropriate business transaction entry points. The entry point to a business transaction is an operation that begins and ends every time the user request is invoked.
- Name the business transactions so that it is recognizable to everyone who monitors your application.

You can keep fine tuning your business transaction configuration, excluding some transactions that you currently detect and adding others that you have not been detecting, as you learn more about which transactions give you the most useful information about your application.

## Basic Configurations

There are three areas you can configure:

- Entry Points
- Exclude Rules
- Custom Match Rules

### To access business transaction detection configuration

1. From the left navigation pane select **Configure -> Instrumentation**.
2. Click the **Transaction Detection** tab if it is not already selected.
3. Click the detection subtab that corresponds to your environment.
4. Select the tier for which you want to configure the transactions or select the application if you want to configure at the application level.
5. To configure a custom configuration for the tier, select Use Custom Configuration for this tier. For information about inheritance for transaction detection, see [Hierarchical Configuration Model](#).

## Entry Points

See [Web Entry Points](#) and [Messaging Entry Points](#).

## Exclude Rules

Create exclude rules to prevent detection of transactions that match certain criteria.

You might want to do this because AppDynamics is detecting business transactions that you are not interested in monitoring or to stay under the agent and controller limits for detected transactions or to exclude a default entry point so you can substitute a more appropriate entry using a custom match rule. See [Entry Points](#) for information about the default auto-detected entry points and [Custom Match Rules](#) for information about creating your own match rules for transaction detection.

For example, the Weblogic Jax RPC Servlets rule excludes any business transaction with a servlet-type entry point based on a class with a name that starts with "weblogic.wsee.server.servlet".

Exclude Business Transaction Match Rule - Servlet

Name	Weblogic JAX RPC Servlets																																																							
Enabled	<input checked="" type="checkbox"/>																																																							
<input type="button" value="Transaction Match Criteria"/> <input type="button" value="Split Transactions Using Request Data"/> <input type="button" value="Split Transactions Using Payload"/>																																																								
<table border="1"> <tr> <td><input type="checkbox"/></td> <td>Method</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>URI</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>HTTP Parameter (Both GET query parameters and POST parameters can be used)</td> <td><input type="button" value="Check for parameter value"/></td> <td><input type="button" value="Parameter Name"/></td> <td><input type="text"/></td> </tr> <tr> <td></td> <td></td> <td></td> <td><input type="button" value="Value"/></td> <td><input type="button" value="Equals"/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Header</td> <td><input type="button" value="Check for parameter value"/></td> <td><input type="button" value="Parameter Name"/></td> <td><input type="text"/></td> </tr> <tr> <td></td> <td></td> <td></td> <td><input type="button" value="Value"/></td> <td><input type="button" value="Equals"/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Hostname</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Port</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>Class Name</td> <td><input type="button" value="Starts With"/></td> <td><input type="text"/> weblogic.wsee.server.servlet.</td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Servlet Name</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Cookie</td> <td><input type="button" value="Check for cookie existence"/></td> <td><input type="button" value="Cookie Name"/></td> <td><input type="text"/></td> </tr> </table>		<input type="checkbox"/>	Method	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input type="checkbox"/>	URI	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input type="checkbox"/>	HTTP Parameter (Both GET query parameters and POST parameters can be used)	<input type="button" value="Check for parameter value"/>	<input type="button" value="Parameter Name"/>	<input type="text"/>				<input type="button" value="Value"/>	<input type="button" value="Equals"/>	<input type="checkbox"/>	Header	<input type="button" value="Check for parameter value"/>	<input type="button" value="Parameter Name"/>	<input type="text"/>				<input type="button" value="Value"/>	<input type="button" value="Equals"/>	<input type="checkbox"/>	Hostname	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input type="checkbox"/>	Port	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input checked="" type="checkbox"/>	Class Name	<input type="button" value="Starts With"/>	<input type="text"/> weblogic.wsee.server.servlet.	<input type="button" value="..."/>	<input type="checkbox"/>	Servlet Name	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input type="checkbox"/>	Cookie	<input type="button" value="Check for cookie existence"/>	<input type="button" value="Cookie Name"/>	<input type="text"/>
<input type="checkbox"/>	Method	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																																				
<input type="checkbox"/>	URI	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																																				
<input type="checkbox"/>	HTTP Parameter (Both GET query parameters and POST parameters can be used)	<input type="button" value="Check for parameter value"/>	<input type="button" value="Parameter Name"/>	<input type="text"/>																																																				
			<input type="button" value="Value"/>	<input type="button" value="Equals"/>																																																				
<input type="checkbox"/>	Header	<input type="button" value="Check for parameter value"/>	<input type="button" value="Parameter Name"/>	<input type="text"/>																																																				
			<input type="button" value="Value"/>	<input type="button" value="Equals"/>																																																				
<input type="checkbox"/>	Hostname	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																																				
<input type="checkbox"/>	Port	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																																				
<input checked="" type="checkbox"/>	Class Name	<input type="button" value="Starts With"/>	<input type="text"/> weblogic.wsee.server.servlet.	<input type="button" value="..."/>																																																				
<input type="checkbox"/>	Servlet Name	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																																				
<input type="checkbox"/>	Cookie	<input type="button" value="Check for cookie existence"/>	<input type="button" value="Cookie Name"/>	<input type="text"/>																																																				
<input type="button" value="Cancel"/> <input type="button" value="Save"/>																																																								

The ASP.NET WebService Session Handler rule excludes any business transaction with an ASP.NET-type entry point based on the System.Web.Services.Protocols.SyncSessionlessHandler class.

Exclude Business Transaction Match Rule - ASP .NET

Name	Service Session Handler																																																		
Enabled	<input checked="" type="checkbox"/>																																																		
<input type="button" value="Transaction Match Criteria"/> <input type="button" value="Split Transactions Using Request Data"/>																																																			
<table border="1"> <tr> <td><input type="checkbox"/></td> <td>Method</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>URI</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>HTTP Parameter (Both GET query parameters and POST parameters can be used)</td> <td><input type="button" value="Check for parameter value"/></td> <td><input type="button" value="Parameter Name"/></td> <td><input type="text"/></td> </tr> <tr> <td></td> <td></td> <td></td> <td><input type="button" value="Value"/></td> <td><input type="button" value="Equals"/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Header</td> <td><input type="button" value="Check for parameter value"/></td> <td><input type="button" value="Parameter Name"/></td> <td><input type="text"/></td> </tr> <tr> <td></td> <td></td> <td></td> <td><input type="button" value="Value"/></td> <td><input type="button" value="Equals"/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Hostname</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Port</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/></td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>Class Name</td> <td><input type="button" value="Equals"/></td> <td><input type="text"/> System.Web.Services.Protocol</td> <td><input type="button" value="..."/></td> </tr> <tr> <td><input type="checkbox"/></td> <td>Cookie</td> <td><input type="button" value="Check for cookie existence"/></td> <td><input type="button" value="Cookie Name"/></td> <td><input type="text"/></td> </tr> </table>		<input type="checkbox"/>	Method	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input type="checkbox"/>	URI	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input type="checkbox"/>	HTTP Parameter (Both GET query parameters and POST parameters can be used)	<input type="button" value="Check for parameter value"/>	<input type="button" value="Parameter Name"/>	<input type="text"/>				<input type="button" value="Value"/>	<input type="button" value="Equals"/>	<input type="checkbox"/>	Header	<input type="button" value="Check for parameter value"/>	<input type="button" value="Parameter Name"/>	<input type="text"/>				<input type="button" value="Value"/>	<input type="button" value="Equals"/>	<input type="checkbox"/>	Hostname	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input type="checkbox"/>	Port	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>	<input checked="" type="checkbox"/>	Class Name	<input type="button" value="Equals"/>	<input type="text"/> System.Web.Services.Protocol	<input type="button" value="..."/>	<input type="checkbox"/>	Cookie	<input type="button" value="Check for cookie existence"/>	<input type="button" value="Cookie Name"/>	<input type="text"/>
<input type="checkbox"/>	Method	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																															
<input type="checkbox"/>	URI	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																															
<input type="checkbox"/>	HTTP Parameter (Both GET query parameters and POST parameters can be used)	<input type="button" value="Check for parameter value"/>	<input type="button" value="Parameter Name"/>	<input type="text"/>																																															
			<input type="button" value="Value"/>	<input type="button" value="Equals"/>																																															
<input type="checkbox"/>	Header	<input type="button" value="Check for parameter value"/>	<input type="button" value="Parameter Name"/>	<input type="text"/>																																															
			<input type="button" value="Value"/>	<input type="button" value="Equals"/>																																															
<input type="checkbox"/>	Hostname	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																															
<input type="checkbox"/>	Port	<input type="button" value="Equals"/>	<input type="text"/>	<input type="button" value="..."/>																																															
<input checked="" type="checkbox"/>	Class Name	<input type="button" value="Equals"/>	<input type="text"/> System.Web.Services.Protocol	<input type="button" value="..."/>																																															
<input type="checkbox"/>	Cookie	<input type="button" value="Check for cookie existence"/>	<input type="button" value="Cookie Name"/>	<input type="text"/>																																															
<input type="button" value="Cancel"/> <input type="button" value="Save"/>																																																			

You can view the other default exclude rules for more examples. See [To View Default Exclude Rules](#).

If you create an exclude rule after transactions based on the exclude criteria have already been created, you need to remove the transactions manually using the Exclude operation from the Actions menu in the business transaction list. See [Business Transactions List Operations](#).

## Default Exclude Rules

AppDynamics provides and enables default exclude rules for Servlets and for ASP.NET.

You can view, modify, disable, or remove these rules.

## To Create Custom Exclude Rules

1. In the Exclude Rules section of the **Transaction Detection** tab, click the Add icon.
2. From the drop-down menu select the Entry Point type for which you want to create the exclude rule and click Next. The New Exclude Business Transaction Match Rule window appears for the selected entry point type.
3. Enter the match criteria for business transactions to be excluded from discovery. The match criteria on which you can configure the rule vary with the entry point type.
4. Click **Create Exclude Rule**.

## To View Default Exclude Rules

1. In the Exclude Rules section of the **Transaction Detection** tab, select an existing exclude rule.

### **Default Exclude Rules for Java Platforms**

Exclude Rules  
If a Transaction matches any of the following rules, it will not be discovered.

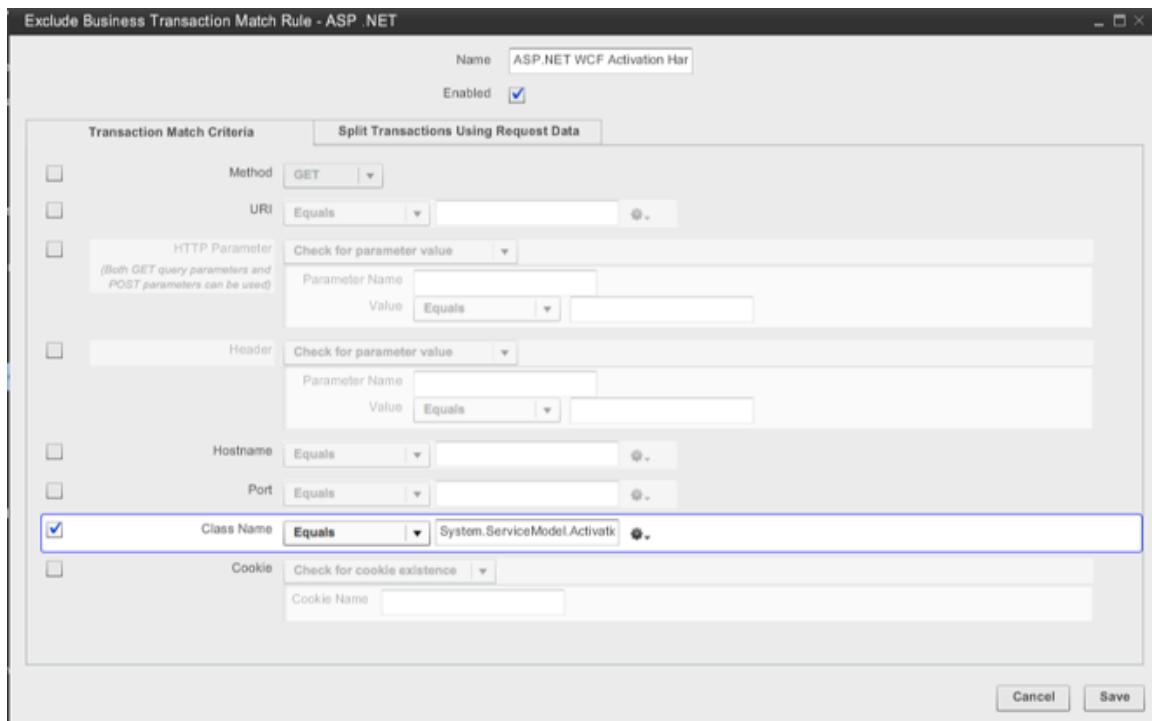
Type	Name	Enabled
Servlet	Apache Axis Servlet	✓
Servlet	Apache Axis2 Servlet	✓
Servlet	Apache Axis2 Admin Servlet	✓
Servlet	Struts Action Servlet	✓
Servlet	Websphere web-services Servlet	✓
Servlet	Websphere web-services axis Servlet	✓
Servlet	JBoss web-services servlet	✓
Servlet	XFire web-services servlet	✓

### **Default Exclude Rules for .NET Platforms**

Exclude Rules  
If a Transaction matches any of the following rules, it will not be discovered.

Type	Name	Enabled
ASP .NET	ASP.NET WCF Activation Handler	✓
ASP .NET	ASP.NET WebService Script Handler	✓
ASP .NET	ASP.NET WebService Session Hand	✓

2. Click the Edit (pencil) icon.
3. To exit the exclude rule configuration screen without changing anything, click **Cancel**.



## Custom Match Rules

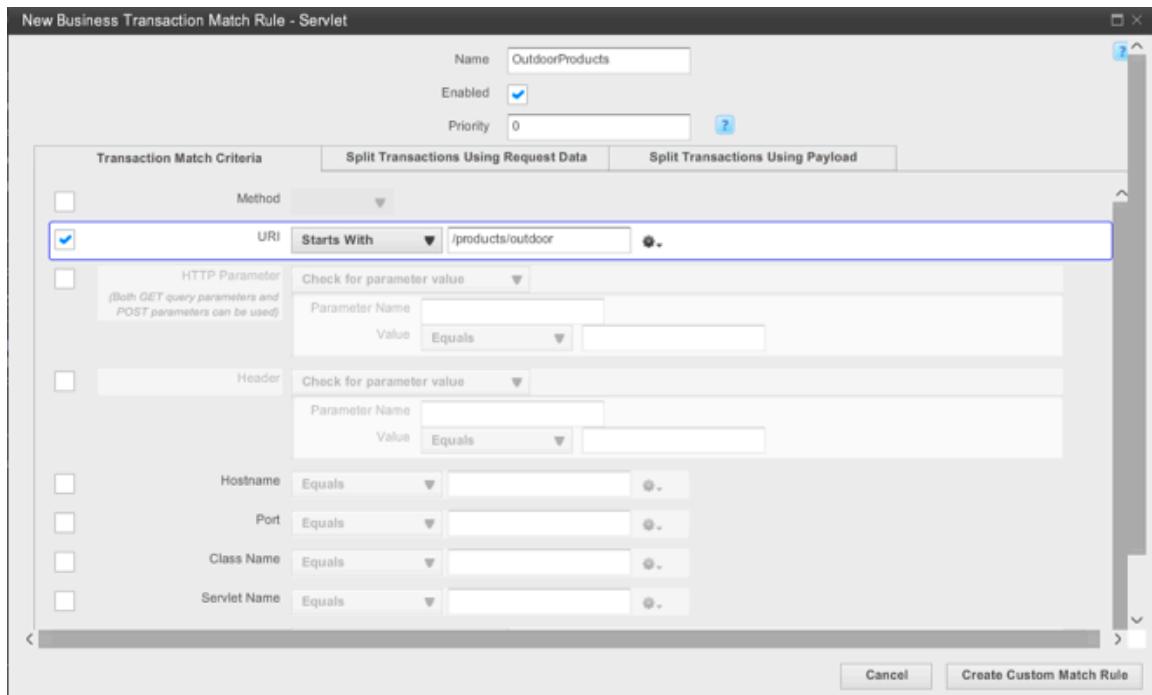
To gain better visibility for selected types of user requests, you can customize how AppDynamics detects business transactions at the tier or application level. You may want to do this if the auto-discovered entry points for certain types do not map precisely to the business perspective or if AppDynamics is detecting too many or too few business transactions.

### To Create Custom Match Rules

1. In the Custom Match Rules section of the **Transaction Detection** tab, click the Add icon.
2. From the drop-down menu select the Entry Point type for which you want to create the custom match rule and click Next. The New Business Transaction Match Rule window appears for the selected entry point type.
3. Check the check boxes and enter the match criteria for AppDynamics to use to detect business transactions of this type. The match criteria on which you can configure the rule vary with the entry point type.
4. Click **Create Custom Match Rule**.

### Sample Custom Match Rule

The following rule creates a custom match rule for a business transaction for servlet-based requests in which the URI begins with "/products/outdoor". This will also be the name of the business transaction as it appears in the business transaction list, unless you rename it. See [Business Transactions List Operations](#) for information about renaming.



## Transaction Splitting

Configuration for some types of custom match and exclude rules allow for transaction splitting. Transaction splitting allows you to fine-tune transaction detection or exclusion based on a parameter or user data.

For example, the following URL represents a Checkout transaction when a user checks out an item from the electronics section:

```
http://nwtrader.com/checkout?category=electronics
```

You can configure the transaction auto-detection mechanism to identify these types of user requests as a Checkout.electronics business transaction by specifying the user request data to use to define the transaction.

With this configuration the business transaction is created dynamically based on the user request. If the request is

```
http://nwtrader.com/checkout?category=clothing
```

the transaction would be Checkout.clothing and so on.

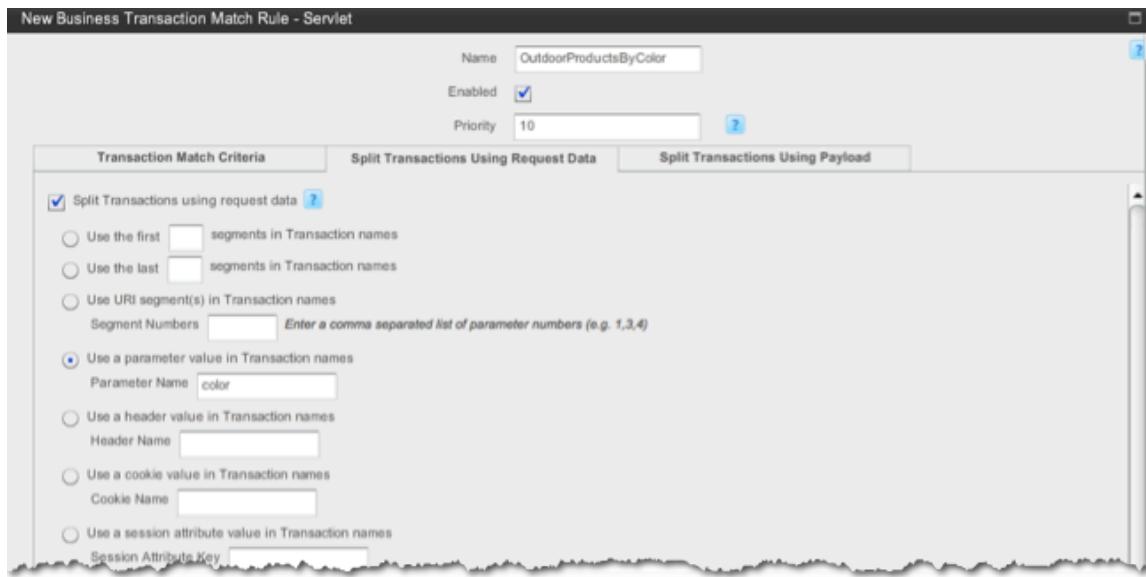
Transaction splitting is optional for the entry point types for which it is offered.

If a splitting part is configured, the transaction is named as:

<custom match rule name> + <name derived from the split configuration>

### Sample Custom Match Rule with Split

This example splits a custom match rule similar to the one created in Sample Custom Match Rule into separate business transactions, one for each color value in the request.



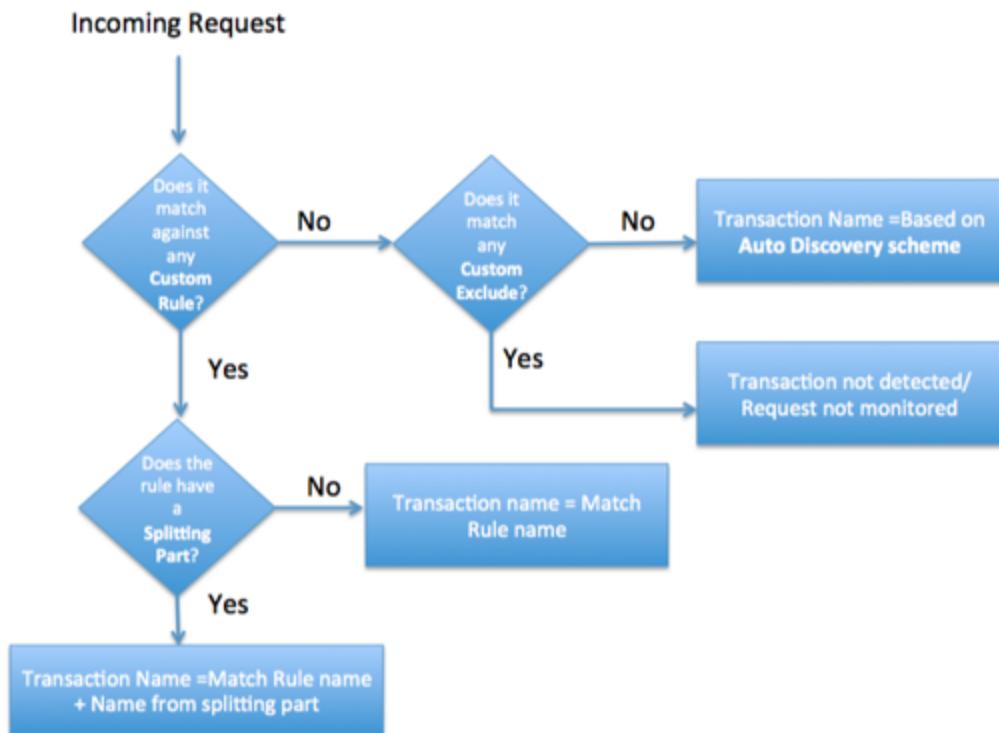
After this configuration is applied, requests that were previously detected as the "/product/outdoor" business transaction are now detected as "/product/outdoor/blue", "/product/outdoor/red" and so on. Now that the "/product/outdoor" business transaction is split, metrics are no longer collected for the old "/product/outdoor" business transaction. Now that the "/product/outdoor" business transaction is split, metrics are no longer collected for the old "/product/outdoor" business transaction.

## Sequence and Precedence for Auto-Naming and Custom Match Rules

AppDynamics transaction detection process uses following sequence for automatically naming the transactions:

1. As soon as the entry point is discovered, AppDynamics checks for a custom match rule. If a custom match rule exists, the business transaction is named based on that rule.
2. If no custom match rule exists, AppDynamics checks the custom exclude rules. If the custom exclude rule exists, the transaction is excluded from discovery.
3. If no custom exclude rule exists, AppDynamics applies its auto-detection naming scheme.

The following illustration shows the sequence for the transaction discovery process:



## The Priority Parameter

You can use the Priority parameter on a custom rule to specify which rule to apply to the detection of a business transaction if it could be detected by more than one rule.

For example, you can set the priority of a custom rule to greater than 0 (zero) if you want the custom rule to take priority over the default rule.

## Transaction Detection Concepts

The following concepts are useful for configuring transaction detection.

- Match Rule Conditions
- Getter Chains in .NET Configurations
- POJOs and POCOs

## Learn More

- Hierarchical Configuration Model
- Business Transactions List
- Web Entry Points
- Messaging Entry Points

## Match Rule Conditions

- Example Match Criteria for a Servlet-based Request
- Learn more

AppDynamics uses match conditions in rules that specify entities to be monitored or excluded from monitoring. You configure match conditions to fine-tune transaction detection, backend detection, data collectors, EUM injection, health rules, etc.

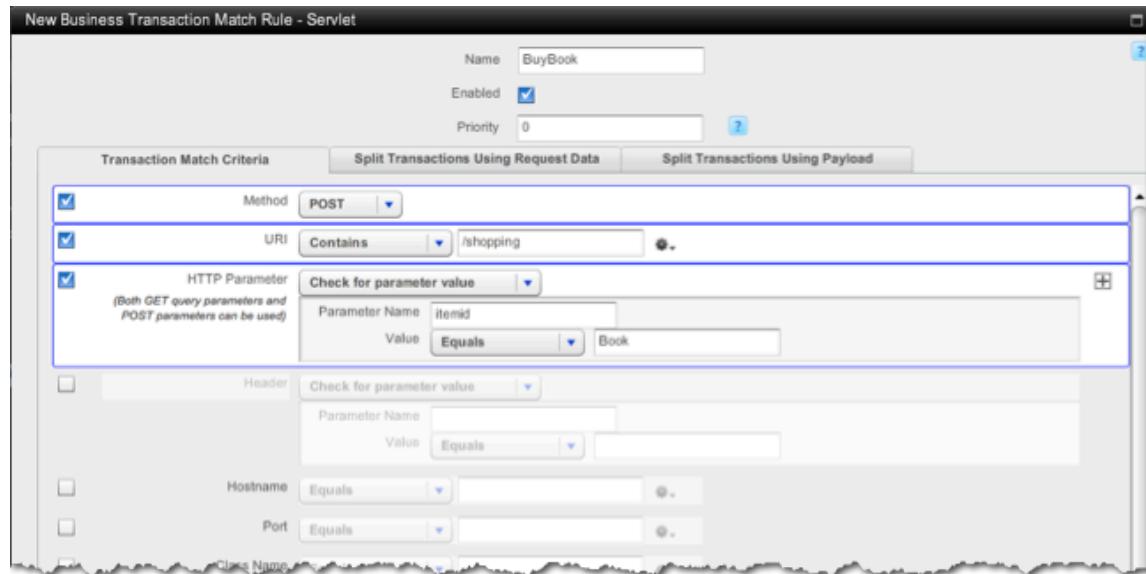
A match condition is a comparison consisting of:

- A match criterion (such as a method name, servlet name, URI, parameter, hostname, etc.)
- A comparison operator typically selected from a drop-down list
- A value

The entities and values being compared and the list of appropriate comparison operators vary depending on the type of configuration.

## Example Match Criteria for a Servlet-based Request

The following example is from a custom match rule named BuyBook used for detecting the business transaction. Detection is based on the discovery of the string "/shopping" in the URI and of a POST parameter with the value "Book" for the itemid parameter. When AppDynamics receives a servlet-based request matching these conditions, it monitors the business transaction.



## Learn more

- Configure Business Transaction Detection

## Regular Expressions In Match Conditions

In AppDynamics you can use regular expressions (sometimes referred to as "RegEx") as an alternative to simple pattern matching. Regular expressions let you combine similar transactions, rather than having to record and identify every variation. You can use regular expressions in custom rules, custom exit points, and other configurations.

AppDynamics uses Java libraries for regular expressions. For more information see:

- Tutorial: <http://download.oracle.com/javase/tutorial/essential/regex/index.html>
- Javadoc: <http://download.oracle.com/javase/1.5.0/docs/api/java/util/regex/Pattern.html>

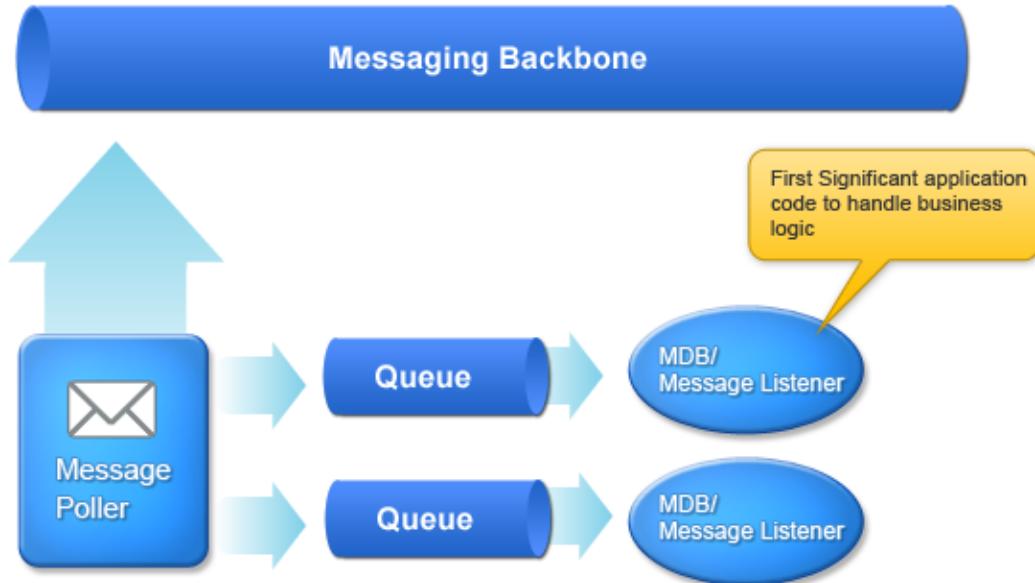
## Messaging Entry Points

- Messaging Entry Points
  - Default Naming Conventions
    - To Access the Default Configuration for Messaging Entry Points
  - Custom Match Rules for Messaging Entry Points
    - Grouping Example
    - Message Payload Example
- Learn More

This topic discusses messaging entry points configuration.

## Messaging Entry Points

When an application uses asynchronous message listeners or message driven beans, such as JMS or equivalent MQ providers as the primary trigger for business processing on the entry point tier AppDynamics can intercept the message listener invocations and track them as business transactions. This is relevant only for the entry point tier.



You can define messaging entry points for queue listeners to monitor Service Level Agreements (SLAs). An SLA is often reflected in the rate of processing by a queue listener. When you monitor a messaging entry point, you can track the rate of processing by a particular queue listener.

### Default Naming Conventions

AppDynamics automatically detects and names the messaging entry points. When a message listener is invoked, the transaction is named after the destination name (the queue name) or the listener class name if the destination name is not available.

### To Access the Default Configuration for Messaging Entry Points

1. Access the business transaction configuration page.  
See [To Access Business Transaction Detection Configuration](#).

2. Scroll down to message entry point entry for your framework.

### Custom Match Rules for Messaging Entry Points

If you want finer control over naming messaging requests, you can use custom match rules either to specify names for your messaging entry points or to group multiple queue invocations into a single business transaction.

See [To Create Custom Match Rules](#) for general information about configuring custom match rules.

### Grouping Example

The following custom match rule groups all transactions to queues starting with the word "Event"

New Business Transaction Match Rule - JMS

Name	EventQueues
Enabled	<input checked="" type="checkbox"/>
Business Transaction Match Criteria	
<input checked="" type="checkbox"/> Message Destination	Queue   Starts With   Event
<input type="checkbox"/> Message Property	Check for property existence   Property Type: BOOLEAN   Property Name:
<input type="checkbox"/> Message Content	Equals   <small>Applies to text messages only</small>
<input type="button" value="Cancel"/> <input type="button" value="Create Custom Match Rule"/>	

## Message Payload Example

The following custom rule uses message properties (headers) to define the custom match rule. You can also use the message content. You can also use message properties or message content to define custom exclude rules to exclude certain messaging entry points from detection.

New Business Transaction Match Rule - JMS

Name	EventQueues
Enabled	<input checked="" type="checkbox"/>
Business Transaction Match Criteria	
<input checked="" type="checkbox"/> Message Destination	Queue   Starts With   Event
<input checked="" type="checkbox"/> Message Property	Check for property value   Property Type: BOOLEAN   Property Name: customerType   Property Value: Priority
<input type="checkbox"/> Message Content	Equals   <small>Applies to text messages only</small>
<input type="button" value="Cancel"/> <input type="button" value="Create Custom Match Rule"/>	

## Learn More

- Configure Business Transaction Detection

# Web Entry Points

- Auto-Detected Entry Points
  - Enable or Disable Transaction Monitoring
  - Enable or Disable Automatic Transaction Detection
- Business Transaction Names
- Learn More

Entry points define where a business transaction begins.

## Auto-Detected Entry Points

AppDynamics classifies entry points by the type of the app server platform and maintains a default detection scheme for each type. The following screenshots show the configurable Java and .NET entry point types and summaries of their default automatic detection and naming rules.

### Auto-Detected Java Entry Points

The screenshot shows the 'Java - Transaction Detection' configuration page. At the top, there are tabs for 'Java - Transaction Detection' and '.NET - Transaction Detection'. Below the tabs are two buttons: 'Copy' and 'Configure all Tiers to use this Configuration'. The main section is titled 'Entry Points' and contains a table with the following rows:

Type	Transaction Monitoring	Automatic Transaction Detection
Servlet	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Servlet requests <a href="#">Configure Naming</a> <input type="checkbox"/> Enable Servlet Filter Detection <a href="#">?</a>
Struts Action	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Struts Action invocations Transactions will be named: ActionName.MethodName
Web Service	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Web Service requests Transactions will be named: ServiceName.OperationName
POJO	<input checked="" type="checkbox"/> Enabled	Any Java method can be the entry point for a Business Transaction. The class to which the method belongs to can be picked using different parameters like its name, its super class name, the interfaces it implements, or the annotations it has.
Spring Bean	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Discover Transactions automatically for all Spring Bean invocations Transactions will be named: BeanName.MethodName
EJB	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Discover Transactions automatically for all EJB invocations Transactions will be named: EJBNName.MethodName
JMS	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all incoming JMS Messages Transactions will be named: Destination Name or Listener Class Name (If Destination Name not available )
Binary Remoting	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Binary Remoting requests (Thrift) Transactions will be named: RemoteInterfaceClassName.methodName

### Auto-Detected .NET Entry Points

[Java - Transaction Detection](#) [.NET - Transaction Detection](#)

[Copy](#) [Configure all Tiers to use this Configuration](#)

▼ Entry Points

Type	Transaction Monitoring	Automatic Transaction Detection
ASP .NET	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for ASP .NET requests <a href="#">Configure Naming</a>
Web Service	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Web Service requests Transactions will be named: ServiceName.OperationName
WCF	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for WCF requests Transactions will be named: ServiceName.OperationName
.NET Class / Method	<input checked="" type="checkbox"/> Enabled	Any .NET method can be the entry point for a Business Transaction. The class to which the method belongs to can be picked using different parameters like its name, its super class name, the interfaces it implements, or the annotations it has.
Message Queues	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for message listeners (IBM XMS, Tibco EMS, Tibco RV, Apache MQ) Transactions will be named: Destination Name or Listener Class Name (If Destination Name not available )

## Enable or Disable Transaction Monitoring

For each entry point type, you can enable and disable transaction monitoring. When monitoring is disabled, the agent stops counting, measuring, recording, etc. all activity on servers of that entry type throughout the application (if detection is being configured at the application level) or for specific tiers (if transaction is being configured at the tier level). Transactions discovered by automatic detection and those discovered by custom rules are equally affected.

For each type you can also enable and disable automatic transaction detection. After you disable detection for an entry point type, no new transactions based on auto detection for that type are discovered. Custom rules are still active and all transactions that are based on custom rules for the entry point type report data. If a transaction was detected earlier when automatic detection was enabled and then you disable it, the agent stops reporting performance metrics for that transaction. However, the transaction is not deleted or excluded.

## Enable or Disable Automatic Transaction Detection

For each entry point type, you can also enable and disable automatic transaction detection.

If automatic detection was enabled and then you disable it, the agent stops reporting metrics for the transactions previously detected by the now disabled entry point and detects only transactions based on custom rules. Calls to methods and operations of the disabled entry point type are no longer auto-detected. Custom rules for the entry point type remain active and the agent reports metrics for them.

## Business Transaction Names

For Java Servlet and ASP .NET transactions you can configure whether to use the entire URI or just part of the URI as the transaction name. You can also name transactions dynamically using a specific part of the request, such as a specific parameter value, cookie value, session attribute value, etc. as the transaction name.

### Servlet Naming Configuration

**Servlet Transaction Naming Configuration**

What part of the URI should be used in the Transaction Name?

Use the full URI  
 Use a part of the URI (for example, if you have dynamic URIs)  
 Use the first ▾ 2 segments of the URI in Transaction Names [What does this do?](#)

Name Transactions dynamically using part of the request

Use URI segment(s) in Transaction names  
 Segment Numbers  Enter a comma separated list of parameter numbers (e.g. 1,3,4)

Use a parameter value in Transaction names  
 Parameter Name  itemid

Use a header value in Transaction names  
 Header Name

Use a cookie value in Transaction names  
 Cookie Name

Use a session attribute value in Transaction names  
 Session Attribute Key

Use the request method (GET/POST/PUT) in Transaction names

Use the request host Transaction in names

Use the request originating address in Transaction names

Apply a custom expression on HttpServletRequest and use the result in Transaction Names [Explain This](#)

[Cancel](#) [Save](#)

### ASP .NET Naming Configuration

**ASP .Net Transaction Naming Configuration**

What part of the URI should be used in the Transaction Name?

Use the full URI  
 Use a part of the URI (for example, if you have dynamic URIs)  
 Use the first ▾ 4 segments of the URI in Transaction Names [What does this do?](#)

Name Transactions dynamically using part of the request

Use URI segment(s) in Transaction names  
 Segment Numbers  Enter a comma separated list of parameter numbers (e.g. 1,3,4)

Use a parameter value in Transaction names  
 Parameter Name

Use a header value in Transaction names  
 Header Name

Use a cookie value in Transaction names  
 Cookie Name

Use a session attribute value in Transaction names  
 Session Attribute Key

Use the request method (GET/POST/PUT) in Transaction names

Use the request host Transaction in names

Use the request originating address in Transaction names

[Cancel](#) [Save](#)

## Learn More

- Hierarchical Configuration Model
- Business Transactions List

- Configure Business Transaction Detection
- Messaging Entry Points

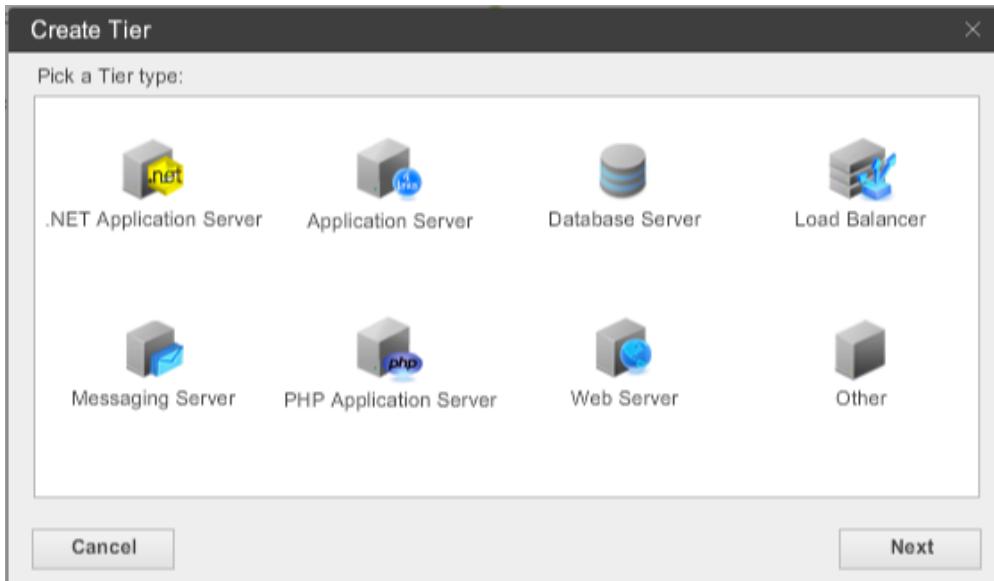
## Creating New Tiers

- Creating a New Tier in an Application
  - To Create a Tier
- Viewing Detailed Information about a Tier or Node
  - To View Detailed Information about a Tier or Node
- Learn More

### Creating a New Tier in an Application

#### To Create a Tier

1. Click **Create Tier**.
2. In the Create Tier window click the tier type.



3. Click **Next**.
4. Enter the name and description of the tier.
5. Click **Finish**.

The tier appears in the left navigation panel under the assigned type. If the menu item for the type does not exist, it is created. For example, if you create a tier of type Other, the tier appears under Other Servers in the left navigation pane.

See [Mapping Application Services to the AppDynamics Model](#) for information about tiers.

### Viewing Detailed Information about a Tier or Node

From the App Server List you can access the Tier and Node Dashboards for specific tiers and nodes.

#### To View Detailed Information about a Tier or Node

1. In the App Server List click the Tree View icon. Alternatively use the left navigation panel and expand the listings.
2. In the listing, select the tier or node that you want to view and click **View Dashboard**.

The dashboard appears. From there you can select the various tabs for details about performance of the tier or node. See [Tier Dashboard](#) or [Node Dashboard](#).

## Learn More

- Logical Model
- Dashboards

# Configure Transaction Snapshots

- To Configure Periodic Snapshot Collection
- To Disable Periodic Snapshots

## To Configure Periodic Snapshot Collection

1. In the left navigation pane click **Configure -> Slow Transaction Thresholds**.
2. In the view navigation, select **Default Thresholds**.
3. Modify the settings for periodic snapshots in the Configure Periodic Snapshot Collection section.

 **Important:** AppDynamics recommends that you do not use low values if you have a high load production environment. When there are thousands or millions of requests per minute, collecting snapshots frequently could flood the system with many snapshots that may not be of high value. Either turn OFF the periodic snapshots and apply to all Business Transactions, or choose a very conservative (high) rate depending on the expected load. For example, if you have high load on the application, choose every 1000th executions or every 20 minutes, depending on the load pattern.

If your SLA-violation-based policies are enabled, you can safely disable the periodic snapshots. See [Policies](#).

## To Disable Periodic Snapshots

1. In the left navigation pane click **Configure -> Slow Transaction Thresholds**.
2. In the view navigation, select **Default Thresholds**.
3. Clear the following check boxes: Take one Snapshot every xxx executions\*...\* and Take one Snapshot every xxx minutes.



4. Click **Save Default Diagnostic Session Settings**.

# Configure Data Collectors

- Configuring Method Invocation Data Collectors
  - To configure Method Invocation Data Collectors
    - To use a getter chain to specify the data collection on method invocation
- Configuring HTTP Data Collectors
  - To Configure HTTP Data Collectors
    - To capture all the parameters for HTTP Request data
    - To use the HTTP content-length header to monitor the correlation between the size of response and overall throughput for an application:
- Configuring SQL Data Collectors
  - To Configure SQL Data Collectors
- Learn More

This topic provides instructions on configuring data collectors.

To access data collector configuration:

1. In the left navigation panel, click **Configure > Instrumentation**.
2. Click the **Diagnostic Data Collectors** tab.

## Configuring Method Invocation Data Collectors

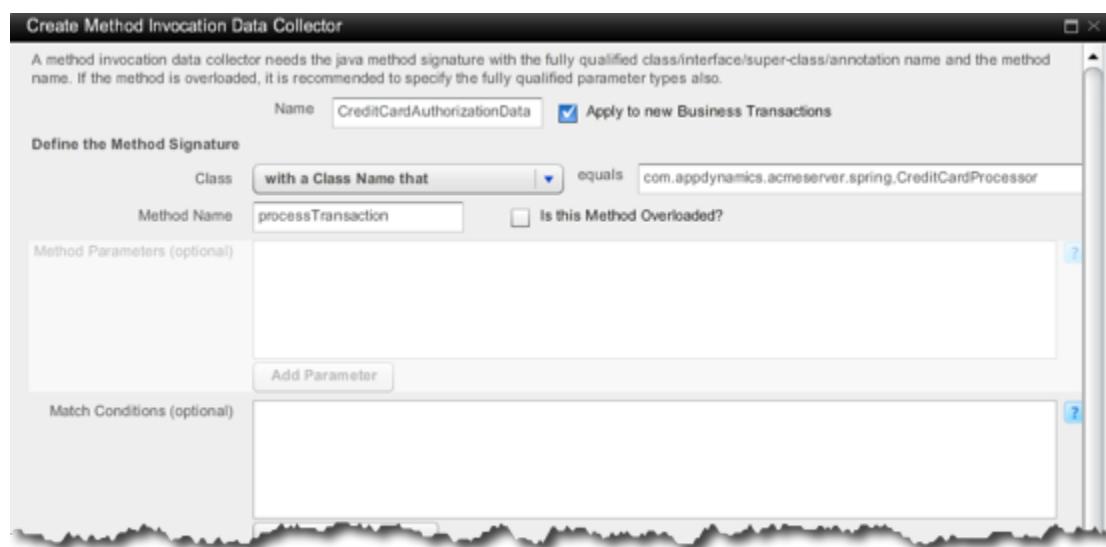
Configure custom method invocation data collectors to capture parameters or return value for a particular method. The captured data appears in panels of the snapshot viewer.

### To configure Method Invocation Data Collectors

1. In the Method Invocation Data Collectors panel, click **Add**.

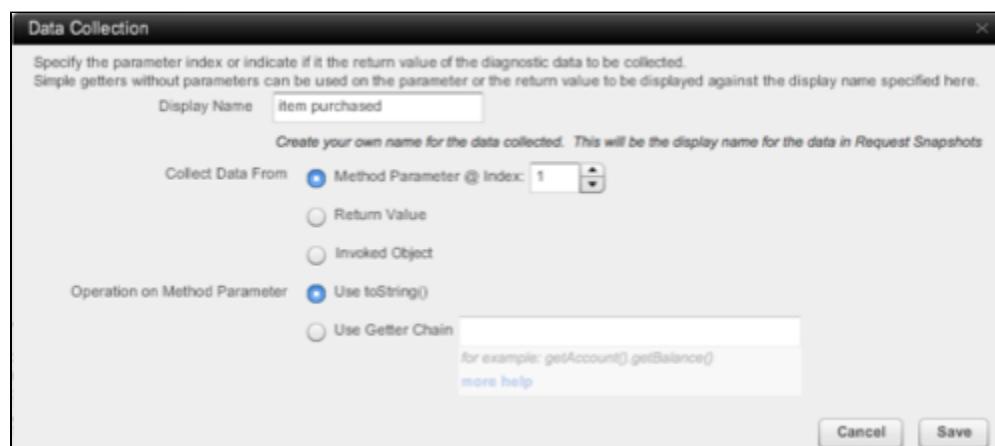
2. Define the method signature.

Provide the class, method, and parameters if the method is overloaded.



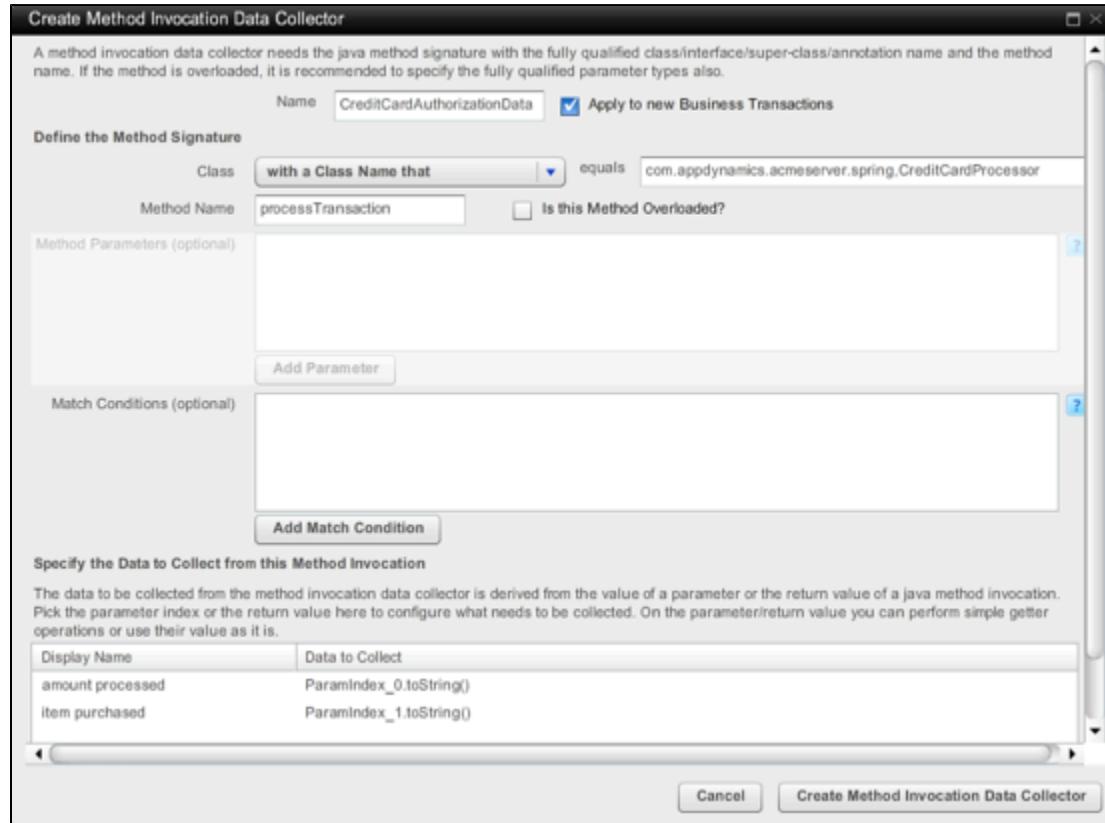
3. To specify the data to collect from this method invocation, click **Add** in the Specify the Data to Collect from this Method Invocation panel.

4. In the Data Collection screen, specify where to collect the data and which operation to use to collect it.



5. To enable this data collector for newly discovered transactions, check Apply to new Business Transactions.

6. Click **Create Method Invocation Data Collector**.



### To use a getter chain to specify the data collection on method invocation

Specify the getter chain in the Data Collection screen.

## Data Collection

Specify the parameter index or indicate if it the return value of the diagnostic data to be collected. Simple getters without parameters can be used on the parameter or the return value to be displayed against the transaction.

Display Name Book Name

Create your own name for the data collected. This will be the display name for the data collector.

Collect Data From

Method Parameter @ Index:

1	▼
▲	▼

Return Value

Invoked Object

Use `toString()`

Use Getter Chain `getTitle().getName()`

for example: `getAccount().getBalance()`



### Specify the Data to Collect from this Method Invocation

The data to be collected from the method invocation data collector is derived from the value of a parameter or the return value of the method. Pick the parameter index or the return value here to configure what needs to be collected. On the right, you can see the resulting display name for the data collector.

Display Name

Data to Collect

Book Name

ParamIndex\_1.getTitle().getName()

See:

[37BKUP: Getter Chains in Java Configurations]

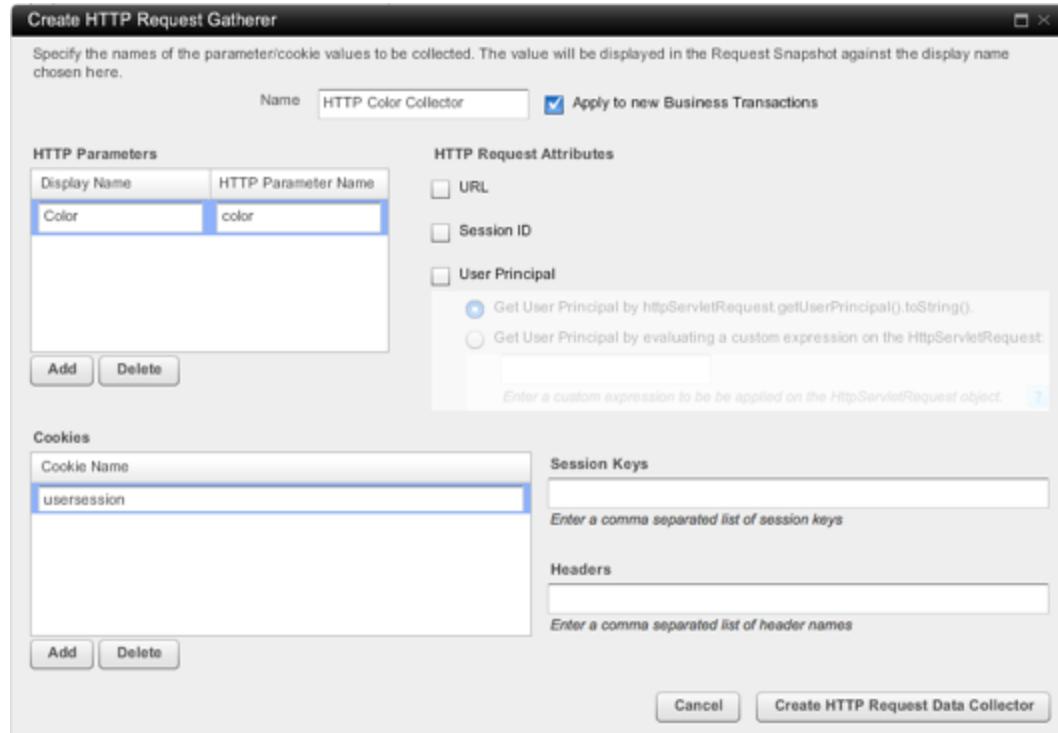
Getter Chains in .NET Configurations

## Configuring HTTP Data Collectors

Configure custom HTTP data collectors to collect HTTP payload data for diagnosis in the transaction snapshot.

### To Configure HTTP Data Collectors

1. In the HTTP Request Data Collectors panel, click **Add**.
2. Click **Add** in the HTTP Parameters section to set the parameter values that you want to collect.
3. Click **Add** in the Cookies section to set the cookie values that you want to collect.
4. If you want to enable this data collector for newly discovered transactions, check **Apply to new Business Transactions**.
5. Click **Create HTTP Request Data Collector**.



You can also configure the HTTP data collector to capture following data:

- URL
- Session ID
- User Principal
- Session Keys
- Headers

### To capture all the parameters for HTTP Request data

Specify "\*" in the HTTP Parameter Name section.



### To use the HTTP content-length header to monitor the correlation between the size of response and overall throughput for an application:

Configure the Content-Length in the Headers section.



The content length header appears in the HTTP Params section of the transaction snapshot.

Name	Value
Header-content-length	56

## Configuring SQL Data Collectors

Use SQL data collectors to restrict diagnostic data captured in transaction snapshots.

### To Configure SQL Data Collectors

1. In the SQL Data Collectors panel click **Add**.
2. In the Create SQL Data Collector screen name and configure the SQL filters.  
You can filter the diagnostic data for SQL queries using either:
  - SQL statement matching
  - SQL statement type (whether this statement is of type INSERT, UPDATE, etc.).
3. To apply this data collector to new business transactions, check **Apply to new Business Transactions**.
4. Click **Create SQL Data Collector**.

The following example shows a SQL data collector to collect only INSERTs that contain the string, "Cart".

Create SQL Data Collector

Filter the queries shown in a Request Snapshot by adding a filter by type of query or query string pattern.

Name: Insert Collector  Apply to new Business Transactions

SQL Statement Matching: Contains  Cart

SQL Statement Type: INSERT

Cancel Create SQL Data Collector

### Learn More

- [Data Collectors](#)

## Configure Thresholds

- Configuring Business Transaction Thresholds
  - [To Configure Slow Transaction Thresholds](#)
  - [Configuring Background Task Thresholds](#)

- Learn More

A threshold is a boundary of acceptable or normal business transaction or background task performance. See [Thresholds](#).

AppDynamics provides default thresholds and you can configure them for your own environment.

To access threshold configuration:

1. Click **Configure -> Slow Transaction Thresholds** in the left navigation pane.
2. Click the **User Transaction Thresholds** or **Background Transactions Thresholds** tab depending on the type of entity for which you want to configure thresholds.

Thresholds for End User Monitoring (EUM) are configured separately. For information about EUM thresholds see [Configuring EUM Thresholds](#).

## Configuring Business Transaction Thresholds

You can configure the thresholds for slow and very slow requests and for stalls. When a transaction or task exceeds a threshold, AppDynamics starts capturing snapshots of it. See [Transaction Snapshots](#). Because the snapshots are not normally captured while performance is within normal range, the snapshots often do not contain the full call graph for the transaction.

You can configure dynamic thresholds using either percentage of standard deviation measurements.

A percentage deviation threshold is based on the moving average of the transaction instances over a certain interval. In the following example, if the average response time is  $x$  milliseconds, a slow transaction would be one which is 50% over  $x$  milliseconds over the last two hours.

User Transaction Thresholds

This section lets you configure Slow Transaction Thresholds, and when to trigger Diagnostic Sessions.

**Slow Transactions Thresholds**

Every Transaction that is processed by the Application will be categorized as normal, slow, very slow, or stalled based on these thresholds.

**Slow Transaction Threshold**

- More than  % slower than the average of the last  hours
- Greater than  Milliseconds
- Greater than  Standard Deviations for the last  hours

**Very Slow Transaction Threshold**

- More than  % slower than the average of the last  hours
- Greater than  Milliseconds
- Greater than  Standard Deviations for the last  hours

**Stall Threshold**

- Disable Stall detection
- Stall occurs when a request takes more than  seconds.
- Stall occurs when a request's response time is  deviations above the average for the last 2 hours.

Apply to all Existing Business Transactions

A standard deviation threshold is also based on a moving average of the transaction instances over a certain interval, but it uses a multiple of the standard deviation rather than a percentage. The following configuration defines a transaction as very slow if it exceeds 4 times the standard deviation over the last three hours.

### Very Slow Transaction Threshold

More than  % slower than the average of the last   ▾

Greater than  Milliseconds

Greater than  Standard Deviations for the last   ▾

You can also configure a static threshold using static values.

## To Configure Slow Transaction Thresholds

1. In the left navigation pane, click **Configure -> Slow Transaction Thresholds**.
2. Click the **User Transaction Thresholds** tab to configure transaction thresholds.
3. In the thresholds tree list, select the scope of the threshold, either:
  - Default Thresholds for all business transactions  
or
  - Individual Business Transaction
4. In the thresholds section, configure the thresholds by selecting the radio button for the type of threshold (dynamic based on percentage, dynamic based on standard deviation or static) and entering the threshold values in the text fields. You can also disable stall detection if desired.
5. If you want these thresholds to apply to all Business Transactions already discovered in your application, click **Apply to all Existing Business Transactions**.
6. Click **Save Default Slow Thresholds** (for default thresholds) or **Save Slow Transaction Thresholds** (for individual thresholds).

AppDynamics displays the resulting performance data in the Transaction Scorecard section of the application and business transaction dashboards.

## Configuring Background Task Thresholds

To configure thresholds for background tasks see [Configure Background Tasks](#).

## Learn More

- [Scorecards](#)
- [Thresholds](#)

## Configure End User Experience

- [Accessing EUM Configuration](#)
  - [To access EUM Configuration](#)
- [EUM Prerequisites](#)
- [Enable and Disable EUM](#)
  - [If you are prompted for the license key](#)
- [Inject the JavaScript Agent for EUM into Your Application](#)
- [Additional EUM Configurations](#)
- [Integration with BMC](#)
- [Learn More](#)

You must explicitly enable End User Monitoring (EUM) at the application level. By default it is disabled. See [Enable and Disable EUM](#).

A special EUM license key is required. See [EUM License](#).

AppDynamics collects metrics from your end users' experience in their Web browsers using a special JavaScript for agent for EUM. This agent must be injected into the Web pages to be monitored. See [Inject the JavaScript Agent for EUM](#).

## Accessing EUM Configuration

### To access EUM Configuration

1. In the left navigation menu, click **Configure -> Instrumentation**.
2. Click the End User Experience tab.

When you are configuring EUM settings, click **Save** in the EUM configuration screens whenever you make a change.

## EUM Prerequisites

To turn on EUM functionality you need to:

- Enable End User Monitoring
- Enter the EUM License Key
- Inject the JavaScript Agent for EUM into your application

## Enable and Disable EUM

In the Javascript Instrumentation tab in the EUM configuration screen:

- To enable EUM for the application, check the Enable End User Experience Monitoring check box.
- To disable EUM for the application, clear the Enable End User Experience Monitoring check box.

Click **Save** in the EUM configuration screen after you have enabled or disabled EUM.

If you first enable then disable EUM, you may see the EUM screen but there will be no EUM data.

The screenshot shows the AppDynamics configuration interface for EUM. The top navigation bar includes 'Flight Search', 'Configure', and 'Instrumentation'. Under 'Instrumentation', tabs for 'Transaction Detection', 'Backend Detection', 'End User Experience' (which is selected), 'Error Detection', and 'Diagnostic Data Collectors' are visible. The 'End User Experience' tab has sub-sections: 'Javascript Instrumentation' (selected), 'Page Naming, Error Detection, Thresholds, etc.', and 'License Key'. A 'Save' button is located in the top right of this section. Below, a note says 'Instrument your HTML pages with the AppDynamics JavaScript Agent' with two steps: 1. Download the JavaScript Agent and 2. Include this line in your pages immediately after the <head> tag. The code example is: <script>window["adrum-start-time"] = new Date().getTime();</script><script src="/adrum.js"></script>. Another 'Save' button is at the bottom.

## If you are prompted for the license key

1. Click the **License Key** tab.
  2. Enter the EUM license key that you received by email from AppDynamics.  
If you do not have your license key, see [EUM License](#) for information about how to obtain it.
  3. Click **Save**.
- On .NET only, the .NET app agent must also be enabled for EUM for full functionality. See [Configure the .NET App Agent for EUM](#).

## Inject the JavaScript Agent for EUM into Your Application

The JavaScript Agent for EUM collects EUM metrics. See [EUM Metrics](#).

The JavaScript Agent for EUM must be injected into the headers of the pages for which you want to see these metrics. There are several ways to accomplish this. See [Inject the JavaScript Agent for EUM](#).

## Additional EUM Configurations

You can also configure:

- [Page Identification and Naming](#)
- [JavaScript and Ajax Error Detection](#)
- [EUM Thresholds](#)
- [EUM Snapshot Collection](#)
- [EUM Deployment](#)

## Integration with BMC

AppDynamics End User Experience is not available with AppDynamics for BMC Software Solutions. For information about integrating AppDynamics with BMC's EUM solution, see [Integrate AppDynamics with BMC End User Experience Management](#)

## Learn More

- [EUM License](#)
- [Inject the JavaScript Agent for EUM](#)
- [Configure the .NET App Agent for EUM](#)
- [Browser Snapshots](#)

## Configure the .NET App Agent for EUM

On .NET only, the .NET app agent must also be enabled for EUM for full functionality.

If, when you set up your .NET app agent, you chose to configure the tiers manually, in the app agent configuration screen you will see an End User Monitoring check box for every tier that you have assigned to the application. Check the check box to enable EUM in every tier for which you want to capture EUM metrics.

The screenshot shows the 'Assign IIS applications to tiers' configuration screen. On the left, a sidebar lists 'Tiers' including Machine Agent, OrderSvc, Portal, QueueSvc, and Weblog. Below this is an 'Add Tier' input field. In the center, a tree view shows 'Local machine IIS applications' under 'IIS Application'. The 'Tier' column maps these to the 'NwTraders' application. The 'End User Monitoring' column contains checkboxes, with the one for 'Portal' being checked and highlighted by a red arrow.

IIS Application	AppDynamics App	Tier	End User Monitoring
Default Web Site	NwTraders	Portal	<input checked="" type="checkbox"/>
aspnet_client			
UtilityApp			
nopCommerce			
Administration			
App_Data			
aspnet_client			
bin			
Content			
Plugins			
Scripts			
Themes			
Views			
Order	NwTraders	OrderSvc	<input checked="" type="checkbox"/>
Queue	NwTraders	QueueSvc	<input checked="" type="checkbox"/>
UtilityApp			
Weblog	NwTraders	Weblog	<input checked="" type="checkbox"/>

If you chose to let AppDynamics configure the tiers automatically, by default all the tiers are configured to enable EUM.

See [Configure the App Agent for .NET](#).

The following features are available when you enable EUM in the .NET configuration:

- More options for injecting the JavaScript Agent for EUM. These include automatic injection (on ASP.NET and ASPX) and assisted injection-using attribute injection. Without this functionality you can use only manual injection.  
See [Inject the JavaScript Agent for EUM](#).
- Server-side information linked to browser snapshots including name and timing for business transactions associated with browser snapshots and linking browser and transaction snapshots when transaction snapshots are available.  
See [Business Transactions in Browser Snapshots](#).

## Configure Page Identification and Naming

- Logic of Page Naming Rule Evaluation
- Default Page Naming Rules
- Custom Page Naming Rules
- Custom Page Exclude Rules
- [Learn More](#)

You can configure how pages, Ajax requests, and Iframes are named in the page list and page dashboards.

- You can use AppDynamics default naming rule, which you can leave as is or modify.
- You can create custom naming rules to override the default convention.
- You can disable the default naming rule and use only your own custom naming rules.
- You can create custom exclude rules to exclude from monitoring pages that meet certain criteria.

In this topic, the term "pages" includes Iframes and Ajax requests as well as base pages.

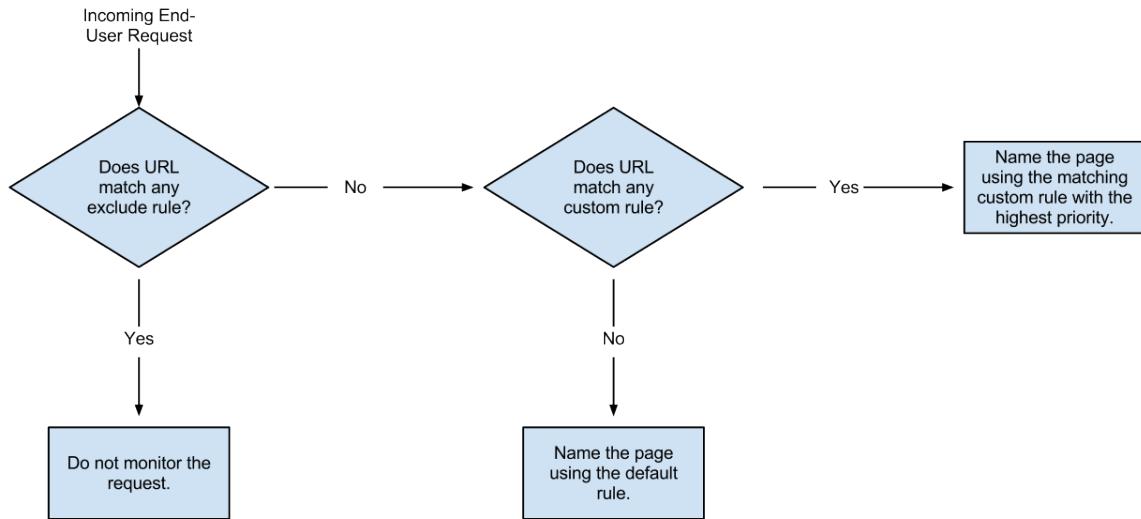
No matter how the page is named, AppDynamics always reports the page name in lower-case.

- Access the EUM configuration screen if you are not already there. See [To access EUM Configuration](#).
- Click the Page Naming, Error Detection, Thresholds, etc tab.
- Expand **Configure how Pages, AJAX Requests, and Iframes will be named**.

Whenever you configure page naming, click **Save** to save the configuration.

## Logic of Page Naming Rule Evaluation

This is the order in which AppDynamics evaluates the page naming rules.



## Default Page Naming Rules

If you enable the default naming configuration and do not modify it, AppDynamics identifies and names your pages using the first 2 segments of the page URL.

You can modify the default configuration in the Default Naming Configuration section of the Page Naming, Error Detection, Thresholds, etc tab. For example, you can include the protocol or domain in the name, or use different segments of the URL, or run a regular expression on the URL, or include query parameters in the name.

**Default Naming Configuration**

Pages will be named using these configurations.

Enabled	<input checked="" type="checkbox"/>
Show Protocol	<input type="checkbox"/>
Show Domain	<input type="checkbox"/> Show Full Domain <input type="checkbox"/> Show Subdomain
What part of the URL should be used in Page Names	<input checked="" type="radio"/> Use first <input type="text" value="2"/> segments <input type="radio"/> Use last <input type="text" value="1"/> segments <input type="radio"/> Use segment numbers <input type="text"/> <p>(comma separated list, starts at 1) <a href="#">Example</a></p>
	<input type="radio"/> Run regex on URI <input type="text"/> Pick indices from regex output <input type="text"/> (comma separated list of indices from regex output, starting at 0.) <a href="#">Example</a>
Query String Parameters to use in Page Name	<input type="text"/>
Comma separated list of parameter names. The values will be used in Page Names.	

If you do not want to use the default convention at all, disable it by clearing the Enabled check box. In this case you must configure at least one custom page naming rule so that AppDynamics can identify and name pages.

## Custom Page Naming Rules

You can create custom rules for identifying and naming pages.

To create a custom page naming rule, click the add icon in the Custom Naming Rules section. Then configure the custom rule for AppDynamics to use to identify and name the page.

This configuration screen is similar to the default configuration screen but it includes a priority field. The priority specifies which rule to apply to the naming of a page if it could be identified by more than one rule. For example, if CustomRuleA specifies **Use the first 3**

**segments of the URL** and has a priority of 9 and CustomRuleB specifies **Use the last 3 segments of the URL** and has a priority of 8, a page in which the URI has more than 3 segments will be named by CustomRuleA because it has a higher priority.

The default rule, if enabled, has a priority of 0.

For example, AppDynamics would use the TicketRule configured below to name the page in AppDynamics when the URL contains "getAllTickets". Otherwise it would use the default rule, or another rule that matched the url with a priority greater than 4 if such a rule exists.

The screenshot shows the 'TicketRule' configuration dialog. It includes fields for Name (TicketRule), Enabled (checked), Priority (4), URL (Contains getAllTickets), Show Protocol (unchecked), Show Domain (Show Full Domain checked), and Query String Parameters to use in Page Name (empty). The 'What part of the URL should be used in Page Names' section is expanded, showing options for 'Use first' (2 segments), 'Use last' (1 segment), and 'Use segment numbers' (1,3,4). The 'Run regex on URI' and 'Pick indices from regex output' fields are collapsed. At the bottom are 'Cancel' and 'OK' buttons.

## Custom Page Exclude Rules

You can configure custom exclude rules for pages. Any page with a URL matching the configuration is excluded from monitoring.

The screenshot shows the 'Custom Exclude Rule' configuration dialog. It includes fields for Name (ExcludeRule1), Enabled (checked), Priority (5), URL (Contains appdynamics), and Query String Parameters to use in Page Name (empty). At the bottom are 'Cancel' and 'OK' buttons.

## Learn More

- Configure End User Experience
- Page List
- Page Dashboard
- Configure Page Names of Any Format

## Configure JavaScript and Ajax Error Detection

- To access JavaScript and Ajax error detection configuration
- Enabling and Disabling EUM Error Detection
- Configuring Rules to Ignore Errors by Script or Error Message
  - To configure Ignore JavaScript Error Rules by Script
- Configuring Rules to Ignore Errors by Page
  - To configure Ignore Error Rules by Page Name
- Configuring Rules to Ignore Errors by URL
  - To configure Ignore Error Rules by URL
- Learn More

You can enable and disable reporting of JavaScript and AJAX request errors.

You can configure which errors are included in the error count by specifying which error to "ignore". AppDynamics does not really ignore "ignored errors". It still reports them, but does not increment the error count for them where error totals are reported.

When enabled, JavaScript and AJAX request errors are reported throughout the EUM UI: in the geo page, browser and device dashboards, in the page list, and in the browser snapshots.

You can specify errors to ignore:

- by script and / or error message
- by page
- by URL

## To access JavaScript and Ajax error detection configuration

1. Access the EUM configuration screen if you are not already there. See [To access EUM Configuration](#).
2. Click the Page Naming, Error Detection, Thresholds, etc tab.
3. Expand **Configure Detection of JavaScript and AJAX Errors**.

Whenever you configure error detection, click **Save** to save the configuration.

## Enabling and Disabling EUM Error Detection

In the EUM error detection configuration screen:

- To enable reporting of JavaScript errors, check the Enable JavaScript Error Capture check box.
- To stop reporting of JavaScript errors, clear the Enable JavaScript Error Capture check box.
- To enable reporting of Ajax errors, check the Enable Ajax Request Error Capture check box.
- To stop reporting of Ajax errors, clear the Enable Ajax Request Error Capture check box.

If both check boxes are clear, AppDynamics will not capture any JavaScript or Ajax request errors from the end-users' Web browsers.

If capture is enabled, you can configure certain errors to be ignored so that they are not counted in the error totals.

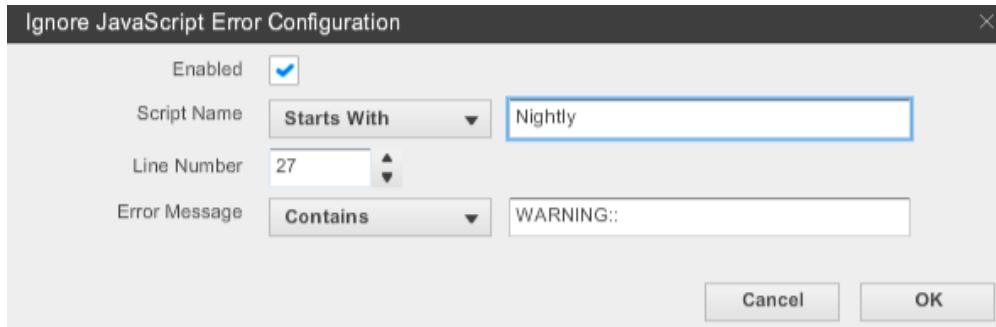
## Configuring Rules to Ignore Errors by Script or Error Message

You can configure AppDynamics to ignore JavaScript errors that are identified by:

- a matching string pattern in the name of the script that generated the error
- line number in the script
- a matching string pattern in the error message

You can specify one, two or all three of these criteria. The more criteria you configure, the more specific is the error to ignore.

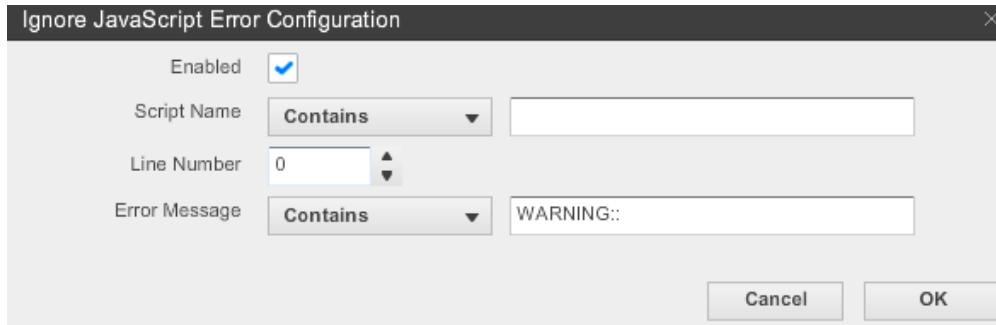
For example, the following configuration, in which all three fields are specified, means "Ignore all errors generated by line 27 of a script whose name starts with "Nightly" and whose error message contains the string "WARNING::".



If the line number were not specified (e.g. set to 0), the configuration would mean "Ignore all errors generated any line of a script whose name starts with "Nightly" and whose error message contains the string "WARNING::".

If neither the line number nor the error message field were specified, the configuration would mean "Ignore all errors generated by any line of a script whose name starts with "Nightly".

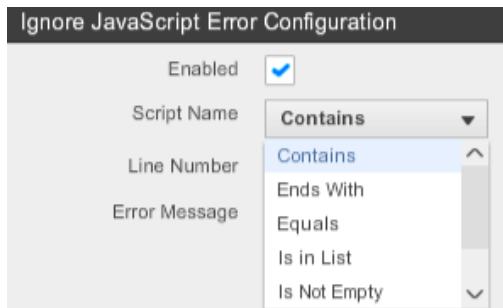
If the error message were the only field specified, the configuration would mean "Ignore all errors generated by any script when the error message contains the string "WARNING::".



## To configure Ignore JavaScript Error Rules by Script

1. Access the configuration screen using the instructions in [To access JavaScript and Ajax error detection configuration](#).
2. In the Ignore JavaScript Error Rules section, click the add icon.
3. Check the Enabled check box to enable the ignore rule.
4. Do one or more of the following depending on how narrowly you want to define the rule:

- Use the dropdown menu to specify the script name of the script generating the error to ignore.



- Specify the line number in the script that generates the error to ignore.
- Specify a string in the error message of the error to ignore, using the dropdown menu.

5. Click **OK**.

To modify an existing ignore rule, select the rule in the list and click the edit icon.  
To remove an ignore rule, select the rule in the list and click the delete icon.

## Configuring Rules to Ignore Errors by Page

You can configure AppDynamics to ignore all errors generated by a specific page, Iframe, or Ajax request. Configure one rule for every page for which you want to ignore all errors.

The screenshot shows a configuration interface with a header 'Configuration'. Below it, there are two rows for defining rules. The first row has 'Page Name' selected, 'Contains' as the operator, and 'process' as the value. The second row has 'Page Name' selected, 'Contains' as the operator, and 'cookie' as the value. A dropdown menu on the right lists operators: Contains, Ends With, Equals, Is in List, and Is Not Empty. A green plus icon is at the top left, and a red minus icon is at the top right.

### To configure Ignore Error Rules by Page Name

1. Access the configuration screen using the instructions in [To access JavaScript and Ajax error detection configuration](#).
2. In the Ignore Page Name Rules section, click the add icon.
3. Use the dropdown menu to specify the page name of the page generating errors that you want to ignore.

Errors from the specified pages will not be counted in the error totals.

To remove an ignore rule, select it in the list and click the delete icon.

## Configuring Rules to Ignore Errors by URL

You can configure AppDynamics to ignore all errors generated by a specific URL. Configure one rule for every URL for which you want to ignore all errors.

The screenshot shows a configuration interface with a header 'Configuration'. Below it, there is one row for defining a rule. The row has 'URL' selected, 'Ends With' as the operator, and 'cookie/noname.html' as the value. A dropdown menu on the right lists operators: Contains, Ends With, Equals, Is in List, and Is Not Empty. A green plus icon is at the top left, and a red minus icon is at the top right.

### To configure Ignore Error Rules by URL

1. Access the configuration screen using the instructions in [To access JavaScript and Ajax error detection configuration](#).
2. In the Ignore URL Rules section, click the add icon.
3. Use the dropdown menu to specify the URL of the page generating errors that you want to ignore.

Errors from the specified URLs will not be counted in the error totals.

To remove an ignore rule, select it in the list and click the delete icon.

## Learn More

- [Configure End User Experience](#)

- Browser Snapshots

## Configure EUM Thresholds

- To access the EUM threshold settings:
- [Learn More](#)

You can configure the thresholds that define slow, very slow, and stalled end-user requests for browser snapshots.

The screenshot shows a summary section of the EUM configuration interface. It includes the following details:

User Experience	Very Slow
End User Response Time (ms)	1187 ms
IFrame	/Timeline.html
URL	http://mycompany.com/timeline.html

You can define EUM thresholds either

- as a multiple of the standard deviation; for example, "Experience is slow if end user response time is slower than 3 X the standard deviation."
- as a static value; for example, "Experience is stalled if end user response time is slower than 30000 ms."

The default thresholds are:

- Slow = 3 x standard deviation
- Very Slow = 4 x standard deviation
- Stalled = 300000 ms

### To access the EUM threshold settings:

1. Access the EUM configuration screen if you are not already there. See [To access EUM Configuration](#).
2. Click the Page Naming, Error Detection, Thresholds etc tab.
3. Expand **Thresholds for Slow End User Experience**.

The screenshot shows the configuration screen for 'Thresholds for Slow End User Experience'. It includes the following fields:

Threshold Type	Setting	Value
Slow Threshold	Standard Deviation	3
Very Slow Threshold	Standard Deviation	4
Stall Threshold	Static (ms)	30000

4. Do one or more of the following:

- Set the Slow Threshold.
- Set the Very Slow Threshold.
- Set the Stalled threshold.

Use the dropdown menu to indicate whether the threshold is based on the standard deviation or a static value.

Type the values in the fields or select them using the scrollbars.

5. Click **Save**.

## Learn More

- [Browser Snapshots](#)

# Configure Browser Snapshot Collection

- [Learn More](#)

By default, when EUM is enabled, the JavaScript agent for EUM captures browser snapshots every 60 seconds.

You can:

- Enable and disable periodic snapshot collection.
- Enable snapshots to be captured whenever JavaScript or Ajax requests receive HTTP error responses. These are responses with an HTTP code equal to or greater than 400.
- Enable and disable slow snapshot collection. When slow snapshot collection is enabled, AppDynamics starts capturing browser snapshots whenever the End User Response Time is higher than the configured threshold.



Slow snapshot collection is useful for troubleshooting problems in development and quality assurance but not recommended for monitoring applications in production.

If periodic collection and error collection and slow collection are all disabled, the agent does not collect any browser snapshots.

To access the snapshot collection settings:

1. Access the EUM configuration screen if you are not already there. See [To access EUM Configuration](#).
2. Click the Page Naming, Error Detection, Thresholds etc tab.
3. Expand **Event Policy Configuration**.

▼ Event Policy Configuration

Enable Slow Snapshot Collection  
Collect Snapshots when End User Response Time is higher than the threshold

Enable Periodic Snapshot Collection  
Collect a normal Snapshot every minute

Enable Error Snapshot Collection  
Collect Snapshots for Pages and IFrames that have JavaScript Errors and AJAX Requests that have HTTP Error

4. Do one or more of the following:

- To enable periodic collection check the Enable Periodic Snapshot Collection check box.
- To disable periodic collection clear the Enable Periodic Snapshot Collection check box.
- To enable error collection check the Enable Error Snapshot Collection check box.
- To disable error collection clear the Enable Error Snapshot Collection check box.
- To enable slow collection check the Enable Slow Snapshot Collection check box.
- To disable slow collection clear the Slow Periodic Snapshot Collection check box.

5. Click **Save**.

## Learn More

- [Browser Snapshots](#)
- [Configure End User Experience](#)

# Customize Your EUM Deployment

- Configuring Custom Deployments
  - To access optional custom configuration
- Alternate Geo Server Location
  - Download the Geo Server File
  - Configure the Geo Server Location
  - Create the IP Mapping File
  - Set Properties in web.xml
- Alternate JavaScript Agent for EUM Extension Location
- Alternate EUM Data Collector Location
- Learn More

Deployment customizations include:

- alternate geo server location
- alternate JavaScript agent extension location
- alternate EUM data collector location

## Configuring Custom Deployments

### To access optional custom configuration

1. Access the EUM configuration screen if you are not already there. See [To access EUM Configuration](#).
2. Click the JavaScript Instrumentation tab.
3. Expand **Advanced**.
4. Expand **Customize your Deployment**.

## Alternate Geo Server Location

By default, end-users' locations are resolved using public geographic databases. You can host an alternate geo server for your countries, regions, and cities instead of using the default geo server hosted by AppDynamics.

The most common reasons for doing this are:

- You have an intranet application for which you do not want transactions to reference a URL outside your organization's network.
- The accessing browsers are coming from many locations through a single gateway, in which case a single IP address appears to the AppDynamics default geo server, hiding the actual locations of your end users.

To host a custom geo server:

1. Download the Geo Server File
2. Configure the Geo Server location
3. Create the IP Mapping File
4. Set Properties in web.xml

## Download the Geo Server File

Download the GeoServer-2.0.zip file from AppDynamics at

<http://download.appdynamics.com/onpremise/public/latest/GeoServer.zip>

This file contains:

- a geo.war
- local-map.xml

Deploy geo.war in a web container.

## Configure the Geo Server Location

Enter URL, including the context root, of your hosted geo server in the Geo Server URL field in the configuration screen. In the following configuration the context root is "/geo".



The screenshot shows a configuration interface with a text input field labeled "Geo Server URL (Optional)". The URL "http(s)://mygeo.acme.com/geo" is entered into the field. A blue question mark icon is located in the top right corner of the input box.

## Create the IP Mapping File

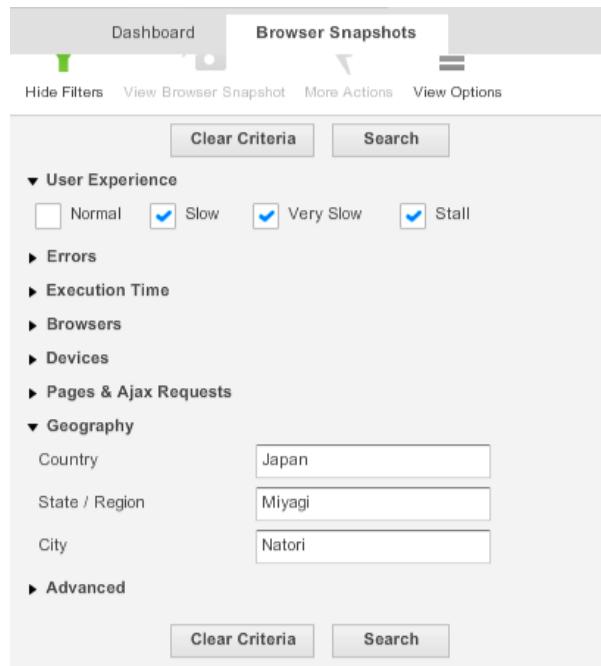
The local-map.xml IP mapping file specifies the locations for which EUM provides geographic data. It maps IP addresses to geographic locations.

Edit the local-map.xml, which was downloaded with the geo.war file, for your environment. This file contains a <location> element for every location to be monitored. The file has the following format.

```
<config>
    <location network="239.0.64.0" subnet-mask="255.255.192.0">
        <country>United States of America</country>
        <region>California</region>
        <city>Mountain View</city>
    </location>
    ... more location entries
</config>
```

The <country>, <region> and <city> elements are required. If the values of <country>, <region> do not correspond to an actual geographic location in the geographic database, map support is not available for the location in the EUM map panel, but EUM metrics are displayed for the location in grid view of the geographic distribution, end user response time panel, trend graphs, browser distribution panel, and in the Metric Browser. The <city> element can be a string that represents the static location of the end-user.

This data is visible in browser snapshots and can be used to filter browser snapshots and to filter browser snapshots for specific locations:



The screenshot shows the EUM 'Browser Snapshots' dashboard. At the top, there are tabs for 'Dashboard' and 'Browser Snapshots'. Below the tabs are buttons for 'Hide Filters', 'View Browser Snapshot', 'More Actions', and 'View Options'. There are two main search buttons: 'Clear Criteria' and 'Search'. On the left, there is a sidebar with expandable sections: 'User Experience' (with checkboxes for Normal, Slow, Very Slow, and Stall), 'Errors', 'Execution Time', 'Browsers', 'Devices', 'Pages & Ajax Requests', and 'Geography'. The 'Geography' section is expanded, showing dropdown menus for 'Country' (Japan), 'State / Region' (Miyagi), and 'City' (Natori). At the bottom of the sidebar are 'Advanced' buttons and 'Clear Criteria' and 'Search' buttons.

The valid names for country and region are those used in the map in the geo dashboard. You can hover over a region in the dashboard to see the exact name (including spelling and case) of the region. See [Geo Dashboard](#).

## Set Properties in web.xml

In the web.xml file, set the ip.mapping.config property to the path of the IP mapping file. The web.xml file is in the geo.war.

You can also set the log directory for the geo server and the number of seconds that geo data should be cached,

Add the mapping information as follows:

```
<init-param>
    <param-name>logs.dir</param-name>
    <param-value>/opt/geo/logs</param-value>
</init-param>
<init-param>
    <param-name>ip.mapping.config</param-name>
    <param-value>/opt/geo/local-map.xml</param-value>
</init-param>
<init-param>
    <param-name>response.cache.seconds</param-name>
    <!-- Default is 1 day. Caching geo info longer than that is bad for mobile
devices. -->
    <param-value>86400</param-value>
</init-param>
```

This example assumes that you are using a modified local-map.xml file. If you created a new mapping file instead, use the name of that file in the <param-value> element instead of "local-map.xml" for the ip.mapping.config property.

## Alternate JavaScript Agent for EUM Extension Location

AppDynamics hosts the JavaScript agent extension on the highly available Amazon CloudFront CDN infrastructure.

To host the JavaScript agent extension, adrum-ext, yourself, download it from the download link in the Customize Your Deployment section of the configuration screen. You will get a version that is compatible with your version of the Controller.

Geo Server URL (Optional) http(s):// (no default)

JavaScript Agent Extension URL (Optional) http(s):// s3-us-west-1.amazonaws.com/jsagent-trunk

[Download adrum-ext.\[version\].js](#)

EUM Data Collector URL (Optional) http(s):// test1-828673714.us-east-1.elb.amazonaws.com

[Additional information](#)

Save it in a Web container and enter the URL of the host in the JavaScript Agent Extension URL field. If you saved the agent file in a directory, for example "js", include the directory name but do not include the name of the actual agent extension file as this may change with subsequent versions. AppDynamics will supply the name of the file when it processes the URL.

JavaScript Agent Extension URL (Optional) http(s):// ec2-184-72-146-37.compute-1.amazonaws.com/js

## Alternate EUM Data Collector Location

The AppDynamics JavaScript agent for EUM reports browser performance data to an EUM data collector. The default data collector stores the data in the Amazon US-WEST Region. Depending on the privacy laws of your country, you may choose a different instance of the EUM data collector.

Contact your AppDynamics sales representative or AppDynamics Support for an appropriate URL for an alternate data collector.

## Learn More

- Inject the JavaScript Agent for EUM
- AppDynamics Support

## Add Information to a Browser Snapshot

- Add User Data
- Modify User Data Size Limit

You can add user information to a browser snapshot. The information appears in a User Data section of the snapshot.

For example you can get the AWS zone or type of node from which the end-user request was served.

## Add User Data

To add user data, add the following method to the pages for which you want the additional data to appear in the browser snapshots.

```
<static> ADRUM.commands ("addUserData", key, value)
```

The results appear in the browser snapshot in a special User Data section.



## Modify User Data Size Limit

The maximum size of all user data in a page is 100 bytes, unless you increase the limit using the `setMaxBeaconLength()` method.

```
<static> setMaxBeaconLength(nbytes)
```

You can set the user data size as high as 2000 bytes. Some browsers will not send packets larger than this, so increasing this value may cause data to be dropped.

If you modify the user data size, the amount of space allocated to all user data fields scales uniformly relative to their default sizes.

## Configure Page Names of Any Format

In the AppDynamics console, you can configure the names of pages, iFrames and Ajax requests based on various parts of the page URL. See [Configure Page Identification and Naming](#).

To use any arbitrary string to name a page, not necessarily some part of the URL, add the `setPageName` method to the page that you want to name.

```
<static> ADRUM.commands ("setPageName", name)
```

The default page name is the DOM document title.

## Handle the window.onerror Event

If any script on your monitored Web pages, including library code, sets the JavaScript window.onerror event, add the following method to the page immediately after setting window.onerror:

```
<static> ADRUM.listenForErrors( )
```

The JavaScript agent for EUM (ADRUM) sets window.onerror to listen for uncaught JavaScript errors. If this listener is overwritten, errors will not be reported.

ADRUM will invoke your original onerror handler.

## Inject the JavaScript Agent for EUM

The JavaScript Agent for EUM allows you to capture EUM metrics and browser snapshots for any web application.

It also can correlate browser snapshots with associated server-side business transaction data for applications that are instrumented by AppDynamics Java and .NET app agents.

### Injection Overview

- Summary of Injection Techniques
- Inject to Capture EUM Metrics and Browser Snapshots
- Inject to Get Server-Side Correlation
  - Getting Full Timing Data for Associated Business Transactions
- Learn More

### Summary of Injection Techniques

The JavaScript Agent for EUM must be injected into all of the web pages for which you want to capture EUM information.

There are several ways to inject the JavaScript Agent for EUM into your web pages. Not all types of injection are supported on all frameworks.

See the Script Injection columns in the [End User Monitoring \(EUM\) Compatibility](#) matrices for information about platforms that support the various types of injection.

- [Manual Injection](#)  
For manual injection, you download the JavaScript Agent for EUM and include it in the head section of the web pages for which you want to collect EUM metrics.

However, this procedure only injects into the head, so it is not guaranteed to return the total execution time for business transactions associated with browser snapshots.

Manual injection is available for all platforms.

- **Automatic Injection**

Automatic injection configuration directs AppDynamics to inject the JavaScript Agent for EUM into the headers and footers on the web pages that you specify for the business transactions that you specify.

Automatic injection is available only for applications that are deployed on Application Servers based on the Apache Jasper JSP compiler for Java platforms or ASP.NET and ASPX for .NET platforms. See the Script Injection column of the matrices in [End User Monitoring \(EUM\) Compatibility](#) to find out if automatic injection is supported for your environment. If it is, it is still possible that automatic injection may not work due to some distinctive characteristics of your web pages. Always try automatic injection thoroughly in a test environment before you use it in production.

For .NET, see also [Configure the .NET App Agent for EUM](#) as configuring the app agent is required to support automatic injection on .NET.

- **Assisted Injection-Using Injection Rules (Java Only)**

Assisted injection using injection rules configuration directs AppDynamics to inject the JavaScript Agent for EUM into the headers and/or footers of pages that you designate through JavaScript injection rules that you create. The JavaScript injection rules specify when and where to inject the agent and the business transactions for which the injection is enabled.

Assisted injection using injection rules is available for Java platforms only.

- **Assisted Injection-Using Attribute Injection**

When enabled through the configuration, the JavaScript Agent for EUM populates attributes in the java.servlet.HttpServletRequest with the HTML to be output in the head section of the page and at the end of the body section.

▼ Advanced Instrumentation of your HTML Pages

Automatic JavaScript Injection      Configure JavaScript Injection

▼ Create Injection Rules

Request Attribute Injection ?

See [Injecting code into the request object](#) for examples of the HTML.  
Attribute injection using attribute injection is available only for applications built on Java servlet or ASP.NET.

## Inject to Capture EUM Metrics and Browser Snapshots

The JavaScript Agent for EUM must be injected into the head sections of the web page for which you want to capture EUM metrics. You can inject the JavaScript Agent for EUM into the page using the following techniques:

- Manual Injection
- Automatic Injection
- Assisted Injection - Using Injection Rules
- Assisted Injection - Using Attribute Injection

## Inject to Get Server-Side Correlation

For servers that are instrumented by an AppDynamics Java or .NET app agent, you can get correlation of browser snapshots with business transaction information on the server side for business transactions that are associated with browser snapshots. When such an association exists, this correlated data appears in the Server Side tab of a browser snapshot:

- business transaction information  
In the Business Transactions section of the Server Side tab is the name and the time of business transactions associated with the browser snapshot.
- snapshot linking  
In the Transaction Snapshots section of the Server Side tab is the list of transaction snapshots that are associated with the browser snapshot, if any transaction snapshots were taken at that time.  
This gives you a continuous view of the user's experience at a certain point in time from the browser through to the backend via the transaction snapshot.

See [Browser Snapshots](#) and [Transaction Snapshots](#) for information about these types of snapshots.

## Getting Full Timing Data for Associated Business Transactions

To ensure that you get the total execution time of a business transaction associated with a browser snapshot, it is necessary to inject the JavaScript Agent for EUM into the footers of the web pages as well as the headers.

You can inject the JavaScript Agent for EUM into the footer of a web page using the following techniques:

- If you use automatic injection for the injecting into the head section, you automatically get injection into the footer also.

- If you use manual injection for the head section, for applications built on Java platforms you can use assisted injection-using injection rules to inject into the footer. Or for applications built on Java servlet or ASP.NET platforms, you can use assisted injection-using attribute injection.

## Learn More

- [Manual Injection](#)
- [Automatic Injection](#)
- [Configure the .NET App Agent for EUM](#)
- [Assisted Injection-Using Injection Rules \(Java Only\)](#)
- [Assisted Injection-Using Attribute Injection](#)
- [Selecting a JavaScript for EUM Instrumentation Strategy](#)

## Manual Injection

- [Download and Include the Agent](#)
  - To access the manual injection panel
  - To inject the JavaScript Agent for EUM
- [Learn More](#)

For manual injection, you download the JavaScript Agent for EUM and include it in the head section of the web pages for which you want to collect EUM data.

However, this procedure only injects into the header, so it does not guarantee complete business transaction timing information when browser snapshots are correlated with server-side business transactions.

## Download and Include the Agent

You configure manual injection from the JavaScript Instrumentation tab of the EUM configuration screen.

### To access the manual injection panel

1. In the left navigation menu, click **Configure -> Instrumentation**.
2. Click the End User Experience tab.
3. Click the JavaScript Instrumentation subtab if it is not already selected.
4. Scroll down to the Instrument your HTML pages with the AppDynamics JavaScript Agent panel.

### To inject the JavaScript Agent for EUM

1. Click **Download the JavaScript Agent**.
2. Click **Save to File** to save it.  
The name of the saved file should be "adrum.js". Save it in the root directory of the page into which you are injecting.
3. To include the JavaScript Agent for EUM in your page, copy the line in the text field in the second step and paste it into the <head> element of the pages that you want to monitor.  
You will get EUM metrics for all pages in which you include this line. If you later decide that you do not want metrics for the page, remove the line.
4. Click **Save** in the configuration screen.

**Save as "adrum.js".**

**Instrument your HTML pages with the AppDynamics JavaScript Agent**

- 1** Download the JavaScript Agent
- 2** Include this line in your pages immediately after the <head> tag:

```
<script>window["adrum-start-time"] = new Date().getTime();</script><script src="/adrum.js"></script>
```

**Path to the downloaded JavaScript agent file**

This inclusion is highly preferable, for convenience, accuracy and maintenance, to copying the entire JavaScript agent into your web pages inline.



The JavaScript for EUM agent is named adrum.js. This script invokes another script called adrum-ext, which performs most of the EUM logic. The adrum-ext script is hosted on Amazon CDN, but you have the option of hosting it at another location. See [Alternate Location for the JavaScript for EUM Agent](#) for information about configuring this option.

If you are using the same JavaScript agent for EUM (adrum.js) for multiple applications, add the following line before the line that includes the agent (adrum.js):

```
window[ "adrum-app-key" ];
```

This is necessary to override the application key set in the AppDynamics Controller.

## Learn More

- [Injection Overview](#)
- [Configure End User Experience](#)
- [Inject the JavaScript Agent for EUM](#)
- [Automatic Injection](#)
- [Assisted Injection-Using Injection Rules \(Java Only\)](#)
- [Assisted Injection-Using Attribute Injection](#)

## Automatic Injection

- To access the automatic injection configuration panel
- [Enable Automatic Injection](#)
  - To enable or disable automatic injection
- [Configuring Automatic Injection](#)
  - To Specify Business Transactions for Automatic Injection
  - To Create Match Rules for Automatic Injection
- [Learn More](#)

One reason to use automatic injection is to inject into the footer of your web pages to capture comprehensive business transaction timing for every end-user request for which there is an associated business transaction on an instrumented server. Automatic injection is one way to inject into the footer.

Another reason to use automatic injection is if you have a large number of web pages to instrument. Automatic injection saves you the trouble of injecting each web page.

Automatic injection is available only for applications built on a Jasper-supported JSP (Java) or ASP.NET or ASPX (.NET) framework. If you plan to use automatic injection and desire correlation with instrumented .NET nodes on the server side, the .NET app agent must be enabled for EUM. See [Configure the .NET App Agent for EUM](#).

**⚠ Warning:** If you configure automatic injection, AppDynamics *strongly* advises you run some traffic through your application *in a test environment* to verify that EUM works and that it is not interfering with your HTML *before you use it in production*.

### To access the automatic injection configuration panel

1. In the left navigation menu, click **Configure -> Instrumentation**.
2. Click the End User Experience tab.
3. Click the JavaScript Instrumentation subtab if it is not already selected.
4. Scroll down to the Advanced panel and expand it if it is closed.
5. Expand **Advanced Instrumentation of your HTML Pages** if it is closed.
6. Click the Automatic JavaScript Injection subtab if it is not already selected.

The screenshot shows the 'Advanced' section of the EUM configuration. Under 'Customize your Deployment', the 'Advanced Instrumentation of your HTML Pages' section is expanded. The 'Automatic JavaScript Injection' subtab is selected. Below it, there is a checkbox labeled 'Enable Automatic Injection of JavaScript' with a help icon next to it. A note below the checkbox states: 'Automatic injection must be enabled for specific Business Transactions. To enable automatic injection, click "Refresh List" and you should see the eligible Business Transactions.'

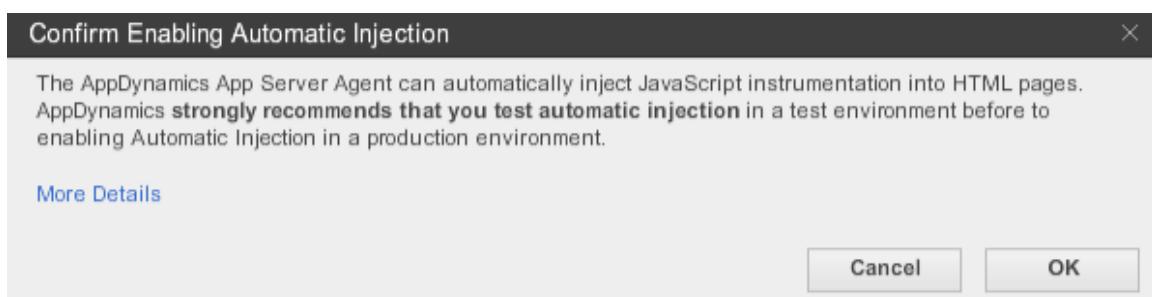
### Enable Automatic Injection

You configure automatic injection from the Advanced Instrumentation of your HTML Pages EUM configuration screen. See [To access the automatic injection configuration panel](#).

#### To enable or disable automatic injection

In the Automatic JavaScript Injection subtab:

1. Do one of the following:
  - To enable automatic injection, check the Enable Automatic Injection of JavaScript check box.
  - To disable automatic injection, clear the Enable Automatic Injection of JavaScript check box.
2. If you enabled automatic injection click **OK** to confirm your action.



## Configuring Automatic Injection

After you have enabled automatic injection:

- You can specify the business transactions for which automatic JavaScript injection is enabled.
- You can qualify which pages to inject, by creating custom match and exclude rules for automatic injection. If you do not configure these rules, by default AppDynamics injects all pages visited by the enabled business transactions.

Use these rules to fine-tune which business transaction to include or exclude from injection based on match criteria. For example, you can exclude all business transactions that have a certain string in their URLs or servlet names or with a certain cookie. The configurations for include rules and exclude rules are similar. It depends on your situation whether it is more convenient to restrict transactions based on inclusion or exclusion.

### To Specify Business Transactions for Automatic Injection

1. Access the Automatic JavaScript Injection subtab in the configuration screen. See [To access the automatic injection configuration panel if you are not there](#).

If automatic injection is enabled, the lists of business transactions for which automatic injection is enabled and disabled are displayed. You can filter the lists by entering a string to filter on in the field above the lists.

It is possible that not all your business transactions will appear in these lists. The lists contain only the business transactions that AppDynamics can parse for automatic injection, those which are Jasper-compiled JSPs or ASPX pages.

2. Specify the business transactions for which injection is automatically enabled by moving them from the Automatic injection possible but not enabled list to the Automatic injection enabled list.

3. Specify the business transactions for which EUM is disabled by moving them from the Automatic injection list to the Automatic injection possible but not enabled list.

4. Click **Save** in the configuration screen..

### To Create Match Rules for Automatic Injection

1. Access the Automatic JavaScript Injection subtab in the configuration screen. See [To access the automatic injection configuration panel if you are not there](#).

2. Expand **Only enable Automatic Injection for certain Pages** if it is closed.

3. To create rules that define which pages to include for automatic injection, click the Add icon in the Request Match Rules section. To create rules that define which pages to exclude from automatic injection, click the Add icon in the Request Exclude Rules section.

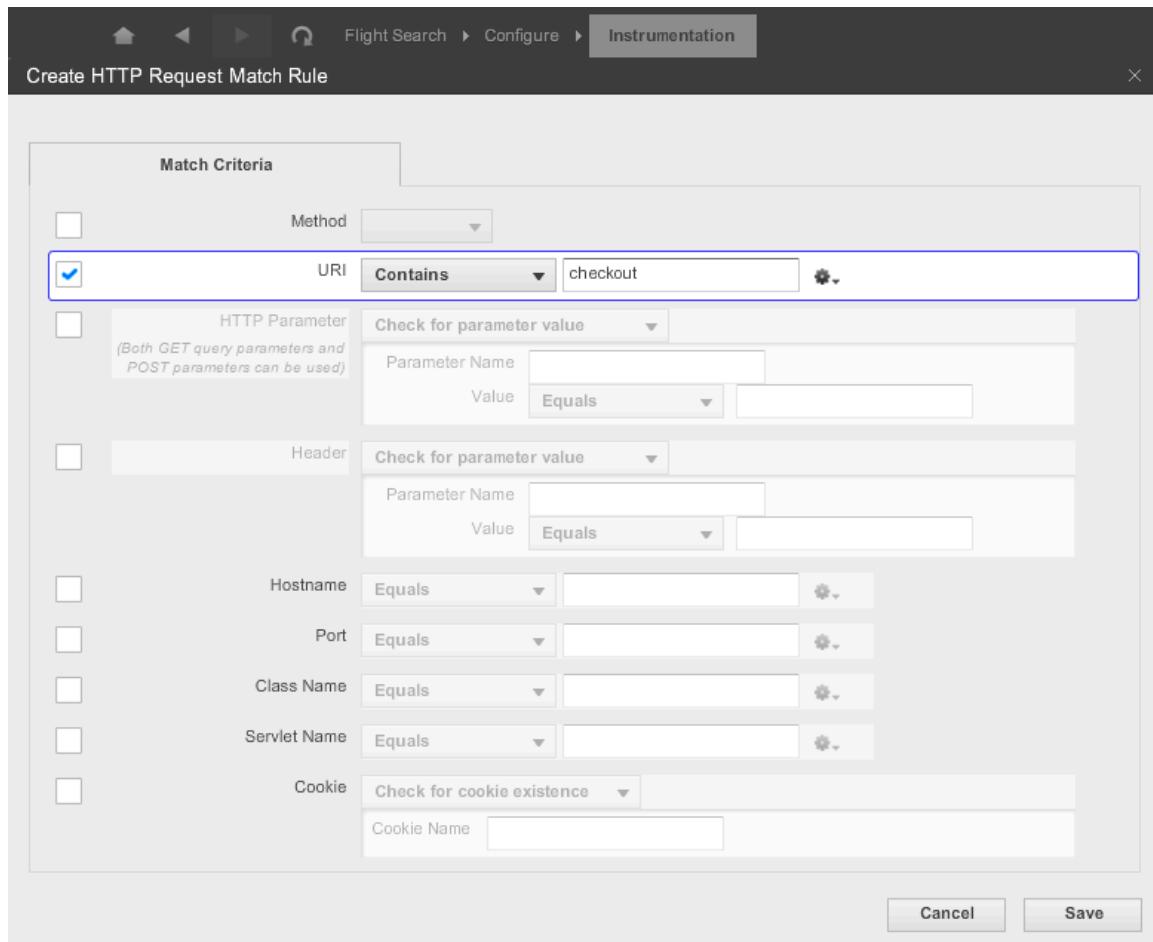
You can later remove these rules by selecting the rule and clicking the delete icon. You can modify them by selecting the edit icon.

4. In the Request Match Rule or Request Exclude Rule screen:

- a. Check the check box next to the criterion to use to include or exclude a request from JavaScript injection.
- b. Configure the match condition for that criterion.

See [Match Rule Conditions](#) for general information about match rules.

Repeat this step for all the criteria that you wish to configure. You can configure multiple criteria, all of which would have to match to qualify the page for inclusion in or exclusion from for injection.



c. Click **Save**.

5. Click **Save** in the outer configuration screen.

You can later edit or remove a match rule by selecting it in the list and clicking the edit or delete icon.

## Learn More

- [Injection Overview](#)
- [Configure the .NET App Agent for EUM](#)
- [Configure End User Experience](#)
- [Inject the JavaScript Agent for EUM](#)
- [Manual Injection](#)
- [Assisted Injection-Using Injection Rules \(Java Only\)](#)
- [Assisted Injection-Using Attribute Injection](#)

## Assisted Injection-Using Injection Rules (Java Only)

- To access the JavaScript injection rules configuration panel
- To create JavaScript injection rules
- [Learn More](#)

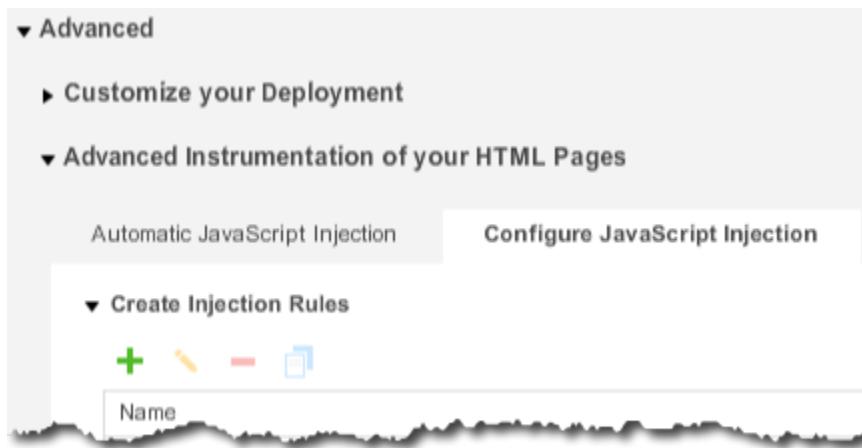
Assisted injection using injection rules configuration directs AppDynamics to inject the JavaScript Agent for EUM into the headers and/or footers of pages that you designate.

You configure JavaScript injection rules that tell AppDynamics when and where to inject the JavaScript and for which the business transactions the injection is enabled.

Assisted injection is available for Java frameworks only.

## To access the JavaScript injection rules configuration panel

1. In the left navigation menu, click **Configure -> Instrumentation**.
2. Click the End User Experience tab.
3. Click the JavaScript Instrumentation subtab if it is not already selected.
4. Scroll down to the Advanced panel and expand it if it is closed.
5. Expand **Advanced Instrumentation of your HTML Pages** if it is closed.
6. Click the Configure JavaScript Injection Rules subtab if it is not already selected.



## To create JavaScript injection rules

1. In the Configure JavaScript Injection Rules tab, expand **Create Injection Rules** if it is closed.
2. Click the add icon.  
The Create Injection Rule screen is displayed.
3. Click the Where to Inject JavaScript tab
4. In the Name field, enter a name for the rule.
5. To enable the rule, check the Enabled check box.  
To disable the rule, clear the Enabled check box.
6. In the Class and method to intercept section, define the match conditions for the class and method that write to the output stream.  
This is the class that AppDynamics intercepts for injection.
7. If the write method is overloaded:
  - a. Check the Is this Method Overloaded? check box.
  - b. Click **Add Parameter**.
  - c. Add the parameters that define the method.
8. In the Pointer to the writer section, select how to obtain the output writer stream in the intercepted method.
9. In the Injection options section, specify:
  - the output stream write method AppDynamics should use to inject
  - when to inject: when the method begins or when the method ends
  - where to inject: before the header or after the footer; to capture full business transaction timing information inject after the footer
  - optional prefix to output before writing the header or footer, such as <DOCTYPE...>
10. Optionally configure the business transactions for which the rule is enabled. By default the rule is enabled for all business transactions. To enable it for specific business transactions only:
  - a. Click the Inject for these Business Transactions tab.

b. Select These Business Transactions.

c. Specify the business transactions for which the injection rule is enabled by moving them from the Other Business Transactions list to the Selected Business Transactions list.

d. Specify the business transactions for which the injection rule is disabled by moving them to (or leaving them in) the Other Business Transactions list.

11. Click **Create Injection rule**.

12. Click **Save**.

You can later edit or remove an injection rule by selecting it in the list and clicking the edit or delete icon.

The following sample injection rule configures injection in the footer at the end of the intercepted method

Name: zk footer  
Enabled:

Class and method to intercept [?](#)

Class: with a Class Name that equals org.zkoss.zk.ui.sys.HtmlPageRenderers  
Method Name: outPageContent  Is this Method Overloaded?

Method Parameters (optional)

Add Parameter

Pointer to the writer [?](#)

Object containing the writer:  Method Parameter @ Index: 2  Return Value  Invoked Object  
Use Getter Chain: this  
for example: getResponse().getWriter()  
[more help](#)

Injection options [?](#)

Output Stream Write Method: write

Inject on:  method begin  method end  
Inject:  eum header  eum footer  
Prefix:

[Cancel](#) [Create Injection Rule](#)

## Learn More

- [Injection Overview](#)
- [Configure End User Experience](#)
- [Inject the JavaScript Agent for EUM](#)
- [Manual Injection](#)
- [Automatic Injection](#)
- [Assisted Injection-Using Attribute Injection](#)

## Assisted Injection-Using Attribute Injection

- [Enabling Attribute Injection](#)
  - [To enable attribute injection](#)
- [Injecting code into the request object](#)
- [Learn More](#)

There are two steps required to set up attribute injection:

- enable attribute injection in AppDynamics
- inject the code

## Enabling Attribute Injection

### To enable attribute injection

1. In the left navigation menu, click **Configure -> Instrumentation**.
2. Click the End User Experience tab.
3. Click the JavaScript Instrumentation subtab if it is not already selected.
4. Scroll down to the **Advanced** panel and expand it if it is closed.
5. Expand **Advanced Instrumentation of your HTML Pages** if it is closed.
6. Click the Configure JavaScript Injection Rules subtab if it is not already selected.
7. Check the Request Attribute Injection check box.

The screenshot shows a configuration interface with a sidebar on the left containing sections like 'Advanced', 'Customize your Deployment', and 'Advanced Instrumentation of your HTML Pages'. Below the sidebar, there are two tabs: 'Automatic JavaScript Injection' and 'Configure JavaScript Injection'. Under 'Configure JavaScript Injection', there is a section titled 'Create Injection Rules' with a checked checkbox labeled 'Request Attribute Injection' and a help icon.

8. Click **Save**.

## Injecting code into the request object

If attribute injection is enabled, you can inject the AppDynamics\_JS\_HEADER and AppDynamics\_JS\_FOOTER objects into the request object. These objects can be accessed in your pages to place the header and footer where they are appropriate.

**Warning:** Be careful about where you inject your JavaScript, as it is possible to break your HTML pages. AppDynamics recommends that you always inject in the <head> element of the HTML and in the footer just before the closing </body> tag.

The following examples show code snippets that can be directly injected into the page:

Manual JSF

```
<h:outputText rendered="#{AppDynamics_JS_HEADER != null}"  
value="#{request.getAttribute('AppDynamics_JS_HEADER')}" escape="false"/>  
<h:outputText rendered="#{AppDynamics_JS_FOOTER != null}"  
value="#{request.getAttribute('AppDynamics_JS_FOOTER')}" escape="false"/>
```

#### Manual JSP

```
<% if (request.getAttribute("AppDynamics_JS_HEADER") != null) { %>
<%=request.getAttribute("AppDynamics_JS_HEADER")%> <% } %>
<% if (request.getAttribute("AppDynamics_JS_FOOTER") != null) { %>
<%=request.getAttribute("AppDynamics_JS_FOOTER")%> <% } %>
```

#### Manual Servlet

```
if (request.getAttribute("AppDynamics_JS_HEADER") != null)
{
    out.write(request.getAttribute("AppDynamics_JS_HEADER"));
}
if (request.getAttribute("AppDynamics_JS_FOOTER") != null)
{
    out.write(request.getAttribute("AppDynamics_JS_FOOTER").toString());
}
```

#### Manual Groovy

```
<g:if test="\${AppDynamics_JS_HEADER}">
    \${AppDynamics_JS_HEADER}
</g:if>

<g:if test="\${AppDynamics_JS_FOOTER}">
    \${AppDynamics_JS_FOOTER}
</g:if>
```

#### Manual Velocity Template

```
#if ($AppDynamics_JS_HEADER)
    $AppDynamics_JS_HEADER
#end
#if ($AppDynamics_JS_FOOTER)
    $AppDynamics_JS_FOOTER
#end
```

#### Manual ASP.NET

```
<% if (Context.Items.Contains("AppDynamics_JS_HEADER"))
    Response.Write(Context.Items["AppDynamics_JS_HEADER"]); %>
<% if (Context.Items.Contains("AppDynamics_JS_FOOTER"))
    Response.Write(Context.Items["AppDynamics_JS_FOOTER"]); %>
```

#### Manual MVC Razor

```
@if(HttpContext.Current.Items.Contains("AppDynamics_JS_HEADER"))  
{ @Html.Raw((string)HttpContext.Current.Items["AppDynamics_JS_HEADER"]) }
```

```
@if(HttpContext.Current.Items.Contains("AppDynamics_JS_FOOTER"))  
{ @Html.Raw((string)HttpContext.Current.Items["AppDynamics_JS_FOOTER"]) }
```

## Learn More

- [Injection Overview](#)
- [Configure End User Experience](#)
- [Inject the JavaScript Agent for EUM](#)
- [Manual Injection](#)
- [Automatic Injection](#)
- [Assisted Injection-Using Injection Rules \(Java Only\)](#)

## Selecting a JavaScript for EUM Instrumentation Strategy

- [Choosing an Injection Method](#)
- [Verifying Injection](#)
- [Reversing Injection](#)
- [Learn More](#)

Injection into the head section of a web page is required to get EUM data for that page. Injection into the footer is optional. See [Getting Full Timing Data for Associated Business Transactions](#) for reasons for injecting into the footers.

## Choosing an Injection Method

If you do not have an obvious preference for injection based on your platform and your requirements, and you do not care about getting the entire business transaction timing information in your browser snapshots, use manual injection to inject into the head section.

Automatic injection requires the least amount of effort because you do not have to instrument every page; however, you must test automatic injection thoroughly before using it in production. Check the matrices at [End User Monitoring \(EUM\) Compatibility](#) to see if automatic injection has been tested in your environment. Note that even if a particular script injection type has been tested on a particular platform, it still may not work for you because of some characteristics of your web pages.

If automatic injection is not an option and you want total business transaction timing data, the you can use manual injection to inject into the head section and assisted injection to inject into the footer. Another alternative is to use assisted injection to inject into both the headers and the footers.

## Verifying Injection

If you configured manual injection and are not seeing EUM metrics after running some load for a while, check the web page to confirm that the JavaScript Agent for EUM is in the page. If it is not, try adding it again one time.

If after two attempts you still do not see EUM metrics, try one of the other injection schemes if they are available for your platform, or call AppDynamics Support.

## Reversing Injection

If you try one way to inject and it does not work, AppDynamics recommends that you undo the current injection configuration before implementing another one.

- To undo automatic injection, just clear the Enable Automatic Injection of JavaScript check box.
- To undo manual and assisted injection using attribute injection, manually delete the JavaScript Agent for EUM from your web pages.
- To undo assisted injection using injection rules, clear the Enable check box for each injection rule in the injection rules list.

If multiple copies of the agent exist on a page, the second copy does not execute.

## Learn More

- [Inject the JavaScript Agent for EUM](#)

# Configure Background Tasks

- Enabling Automatic Discovery for Background Tasks
  - To enable discovery for a background task using a common framework
- Configuring Thresholds for Background Tasks
  - To configure thresholds for all background tasks
  - To configure thresholds for an individual background task
- Re-configuring a Business Transaction as a Background Task
  - To re-configure a business transaction as a background task
- Learn More

Configuring background tasks consists of two basic steps:

1. [Enabling automatic discovery](#) so that AppDynamics will detect the background process and monitor it.
2. Configuring thresholds for what you consider to be slow, very slow, or stalled performance behavior.

Sometimes AppDynamics will discover a background task and interpret it to be a business transaction. You can instruct AppDynamics to [Configuring a Business Transaction as a Background Task](#).

## Enabling Automatic Discovery for Background Tasks

Automatic discovery of background tasks is disabled by default. When you know that there are background tasks in your application environment and you want to monitor them, first enable automatic discovery so that AppDynamics will detect the task.

AppDynamics provides preconfigured support for some common frameworks. If your application is not using one of the default frameworks you can create a custom match rule.

### To enable discovery for a background task using a common framework

1. In the left navigation pane, click **Configure -> Instrumentation**.
2. On the Transaction Detection tab, select the tier for which you want to enable monitoring.
3. Click **Use Custom Configuration for this Tier**.
4. Scroll down to the Custom Match Rules pane.
5. Do one of the following
  - If you are using a pre-configured framework, select the row of the framework and click the pencil icon, or double-click on the row to open the Business Transaction Match Rule window. By default the values are populated with rule name and the class and method names for the particular framework. Verify that those are the correct names for your environment.  
OR
  - If you are using a custom framework, select the match criteria and enter the Class Name and Method Name.

The Background Task check box should be already checked.

6. Check **Enabled**.

7. Click **Save**.

The custom match rule for the background task will take effect and the background task will display in the Business Transaction List.

Once you enable discovery, every background task is identified based on following attributes:

- Implementation class name
- Parameter to the execution method name

For additional details see:

---

---

Configure Background Tasks (Java)

---

---

Not Supported for .NET

---

---

Not Supported for PHP

---

## Configuring Thresholds for Background Tasks

The out-of-the-box stall thresholds for background tasks are disabled. This is because the default configuration for stall detection is 45 seconds, which is usually not enough time to detect stalled background tasks.

AppDynamics recommends that you configure a threshold that is suitable for the background task in your environment.

AppDynamics also recommends that you use static thresholds for slow and very slow background tasks, if they have infrequent load patterns such as once every night. This is because the dynamic moving average-based thresholds are more suitable for production load scenarios and will automatically classify a background process as slow or very slow.

### To configure thresholds for all background tasks

1. In the left navigation pane click **Configure -> Slow Transaction Thresholds**.
2. Click the Background Tasks Thresholds tab.
3. Set the thresholds for all background tasks.
4. If you want these thresholds applied to existing background tasks, check **Apply to all Existing Background Task Business Transactions**.
5. Click **Save Default Background Task Thresholds**.

All new background tasks will use the thresholds that are configured here.

You can override default thresholds for an individual background tasks by configuring individual task thresholds.

### To configure thresholds for an individual background task

1. In the Business Transactions List, right-click on the background task and select **Configure Thresholds**. AppDynamics displays the threshold settings for that background task.
2. Update the thresholds.
3. Save the changes.

## Re-configuring a Business Transaction as a Background Task

You can re-configure an auto-discovered business transaction as a background task.

### To re-configure a business transaction as a background task

1. In the left navigation pane click **Business Transactions**.
2. In the Business Transactions List select the business transaction that you want to mark as a background task.
3. Right-click on the selected business transaction and select **Set as Background Task**.
4. Verify that **Set as Background Task** is selected in the dialog.
5. Click **OK**.

If you discover that a background task is better represented as a regular business transaction, repeat steps 1 and 2 then right-click and select **Set as User Transaction**.

## Learn More

- [Configure Background Tasks \(Java\)](#)

# Configure Backend Detection

- Auto-Discovery
- Backend Discovery and Naming
  - To view the discovery rules for an auto-discovered backend
  - Default Properties for Backend Detection in Java Environments
  - Default Properties for Backend Detection in .NET Environments
- Customizing Backend Discovery Rules
  - To copy an entire backend detection configuration to all tiers
  - To copy an entire backend detection configuration to another tier or application
  - To edit a backend discovery rule
- Creating New Discovery Rules for Auto-Discovered Backends
  - To create a custom discovery rule for an auto-discovered backend type
- Examples
- Learn More

In AppDynamics, databases and remote services are collectively known as backends. Backends are born from exit points. An exit point is an exit call from an instrumented node to an uninstrumented node or other service, such as a message queue. Such an exit call results in backend discovery.

Each type of supported remote service and database has a list of properties associated with it. Each supported backend is identified by its type and the related properties. The set of configured properties is referred to as a backend auto-discovery rule. AppDynamics uses auto-discovery rules to identify and name the databases and remote services.

The preconfigured auto-discovery rules might not always match exactly how you want to measure performance in your specific application. You can change the properties and how they are used to aggregate activity to measure what you choose.

## Auto-Discovery

By default, the configuration for backend detection is inherited from the tier or application level configuration. See [Hierarchical Configuration Model](#).

The auto-discovery rules for backends that can be configured are listed, with their default configurations, in the Automatic Backend Discovery list in the **Backend Detection** tab. There is a different configuration for each auto-discovered backend type.

The default discovery rules for each backend type include the following:

- Whether automatic discovery is enabled
- Whether correlation with the application is enabled
- Properties used to identify and name the backend

Sometimes you might need a different set of discovery rules to meet your specific criteria. In these cases you can create a custom rule that meets those criteria. Or you might want to disable some or all of the default rules and create custom rules for detecting all your backends. AppDynamics provides flexibility for configuring backend detection.

For example, detection of HTTP backends is enabled by default. The backends are identified by the host and port and correlation with the application is enabled. To change the detection for HTTP backends in some way - disable correlation, or omit the port number from the detected name, or use only certain segments of the host in the name, you can edit the automatic discovery rule for the HTTP type. In most cases, you can achieve the level of customization that you need by editing the default rules.

## Backend Discovery and Naming

For each auto-discovered backend type, there is a set of configurable properties that are used to identify the backend. You can change

the default configuration for the backend types shown in the UI.

For example, you can ignore the Vendor property for JMS backend detection. If you do, then all JMS backends with the same destination and destination type would be identified as the same backend and Vendor is ignored for the purposes of identification. The metrics for all JMS backends with the same destination and destination type would be aggregated as one backend.

For properties that you do want to use, you can also configure how AppDynamics should use the property. For example, if the URL property is used to identify a backend and your URLs include a file name, using the default configuration, AppDynamics identifies a separate backend for every file. To reduce the number of backends identified using the URL property, you can configure which parts of the URL to use. For example, you could run a regular expression on the URL to discard the file name, or any other parts of the URL that you do not want used for identification.

You can modify the database and remote service discovery configuration in the following ways:

- Use the default configuration
- Use one or more specified segments of a property
- Run a regular expression on a property
- Execute a method on a property

## To view the discovery rules for an auto-discovered backend

1. Access the backend detection screen using these steps:
  - a. Select the application.
  - b. In the left navigation pane, click **Configure -> Instrumentation**.
  - c. Click the **Backend Detection** tab.
  - d. Select the application and the tab corresponding to the backend platform.  
The Automatic Backend Discovery default configurations are listed.
2. In the Automatic Backend Discovery list, click the configuration to view and a summary of the configuration appears on the right.

For example, the following figure shows that JMS backends are auto-discovered using the Destination, Destination Type, and Vendor.

The screenshot shows the 'Java - Backend Detection' configuration page. On the left, there's a tree view of application tiers: 'AcmeOnlineBookStore' with 'eCommerceServer', 'inventoryServer', and 'orderProcessingServer'. Below the tree is a 'Copy' button and a 'Configure all Tiers to use this Configuration' button. The main area is titled 'Automatic Backend Discovery' for 'JMS'. It lists several backend types: MQ, JDBC, JMS, HTTP, RMI, and Web Service. All are marked as 'Enabled' with green checkmarks. Under 'Automatic Discovery', 'Enabled' is checked, and 'Correlation Enabled' is also checked with a green checkmark. A note says 'Name JMS Backends Using Destination, DestinationType, Vendor'. At the bottom right is a 'Edit Automatic Discovery' button.

## Default Properties for Backend Detection in Java Environments

The following table lists the auto-discovered backend types that are configurable in a Java environment, the available configurable properties and whether the property is used in the default auto-discovery rules.

Type	Configurable Properties	Property Used by Default
		in Detection and Naming
IBM MQ	Destination	Yes
	Destination Type	Yes
	Host	Yes
	Port	Yes

	Major Version	Yes
	Vendor	Yes
HTTP	Host	Yes
	Port	Yes
	URL	No
	Query String	No
JDBC	URL	Yes
	Host	Yes
	Port	Yes
	Database	Yes
	Version	Yes
	Vendor	Yes
JMS	Destination	Yes
	Destination Type	Yes
	Vendor	Yes
Cassandra	Host	Yes
	Port	Yes
	transport	Yes
	keyspace	Yes
RMI	URL	Yes
Binary Remoting	Host	Yes
	Port	Yes
	transport	Yes
Web Service	Service	Yes
	URL	No
	Operation	No
	Soap Action	No
	Vendor	No

## Default Properties for Backend Detection in .NET Environments

The following table lists the auto-discovered backend types configurable in a .NET environment, the available configurable properties and whether the property is used in the default auto-discovery rules.

Type	Configurable Properties	Property Used by Default in Detection and Naming
WCF	Remote Address	Yes
	Operation Contract	No
	URL	No

	Host	No
	Port	No
	Soap Action	No
Queues	Host	No
	Destination	Yes
	Destination Type	No
	Vendor	No
HTTP	Host	No
	Port	No
	URL	Yes
	Query String	No
Web Service	Service	No
	URL	Yes
	Operation	No
	Soap Action	No
ADO.NET	Host	Yes
	Port	No
	Database	Yes
	Vendor	No
	Connection String	No

## Customizing Backend Discovery Rules

You can customize backend visibility to suit your monitoring needs. You can configure the detection of databases and remote services (collectively known as backends) in the following ways:

- [Edit the default discovery rules](#)
- [Create new discovery rules](#)
- [Configure custom exit points to monitor calls and enable detection for backend types that are not auto-discovered.](#)

The precise configurations vary according to the type of the backend, but the following general features are configurable:

- Discovery Enabled - You can enable and disable auto-discovery for the backend type. Undiscovered backends are not monitored.
- Correlation Enabled - You can enable and disable correlation. Enabling correlation lets AppDynamics pass information about the application to and through the backend. If you do not care about activity downstream from the backend, you may want to disable correlation.
- Backend Naming

Custom rules include the following additional settings:

- Name for the custom rule
- Priority - Used to set precedence for custom rules.
- Match Conditions - Used to identify backends that are identified by custom rules. See [Match Rule Conditions](#)

## To copy an entire backend detection configuration to all tiers

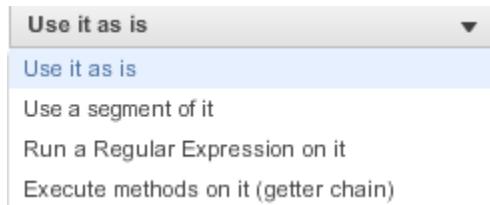
1. Access the backend detection screen. See [To access the backend detection tab](#).
2. In the left panel select the application or tier whose configuration you want to copy.
3. Click **Configure all Tiers to use this Configuration**.

## To copy an entire backend detection configuration to another tier or application

1. Access the backend detection screen. See [To access the backend detection tab](#).
2. In the left panel select the application or tier whose configuration you want to copy.
3. Click **Copy**.
4. In the Application/Tier browser, choose the application or tier to copy the configuration to.
5. Click **OK**.

## To edit a backend discovery rule

1. In the Automatic Backend Discovery list, select the backend type to modify.  
The rule summary appears in the Automatic Discovery panel on the right.
2. Click **Edit Automatic Discovery**.  
The Edit Automatic Backend Discovery Rule window appears
3. For each property that you want to configure:
  - Select the property in the property list.
  - Check the property check box to use the property for detection; clear the check box to omit it.
  - If you are using the property, choose how the property is used from the drop-down list.



4. Check **Enabled** to enable the rule; clear the check box to disable it.
5. Check **Correlation Enabled** to enable correlation.
6. Click **OK**.

## Creating New Discovery Rules for Auto-Discovered Backends

### To create a custom discovery rule for an auto-discovered backend type

1. In the Automatic Backend Discovery list, select the backend type.  
The Custom Discovery Rules editor appears in the right panel below the Automatic Discovery panel.
2. Click **Add** (the + icon) to create a new rule or select an existing rule from the list and click the edit icon to modify one.
3. Enter a name for the custom rule.
4. Enter the priority for the custom rule among all other custom rules for this backend type. The higher the number, the higher the priority. A value of 0 (zero) indicates that the default rule should be used.
5. Configure the match conditions that define which backends.

Match conditions defined on the naming properties are used to identify which backends should use the custom rule. Backends that do not meet all the defined match conditions are discovered according to the default rule.

For example, to use the custom rule for JDBC backends with port numbers that start with 80, you would define the match condition:

Port Starts With 80

The comparison operators for defining the match conditions are: Equals, Starts With, Ends With, Contains, and Matches Regular Expression.

6. Configure the Enabled, Correlation Enabled, and Backend Naming Configuration as described in [To edit a default backend discovery rule](#).

7. Click **OK** to save the configuration.

## Examples

A common use case for changing the backend discovery rules is to combine auto-discovered backends of the same type that share certain properties.

For example, AppDynamics auto-discovers JDBC backends based on host, port, URL, database, version and vendor values. To combine all JDBC databases that contain "EC2" in their host names to be monitored as a single database, follow steps 1 through 4 described above in [To create a custom discovery rule for an automatically discovered backend type](#) and use the following match condition:

Host Contains EC2

Similarly, you can group all Web Service remote services that share the same service value, or all RMI backends that have URLs that match a regular expression, and so on.

## Learn More

- [Configure Custom Exit Points](#)
- [Match Rule Conditions](#)
- [Hierarchical Configuration Model](#)

## Configure Custom Exit Points

- [To create a custom exit point](#)
  - [To split an exit point](#)
  - [To group an exit point](#)
- [To define custom metrics for a custom exit point](#)
- [To define transaction snapshot data collected](#)
- [Learn More](#)

For additional details see:

---

[Configure Custom Exit Points \(Java\)](#)

---

[Configure Custom Exit Points \(.NET\)](#)

---

Not Supported for PHP

---

Custom exit points provide identification for backend types that are not automatically detected, such as file systems, mainframes etc.

For example, you can define a custom exit call to monitor the file system read method. Custom exit points appear as unresolved backends in the flow maps. Unresolved backends are shown on flow maps with this icon .

You define a custom exit point by specifying the class and method used to identify the backend. If the method is overloaded, you need to add the parameters to identify the method uniquely.

You can restrict the method invocations for which you want AppDynamics to collect metrics by specifying match conditions for the method. The match conditions can be based on a parameter or the invoked object.

You can also optionally split the exit point based on a method parameter, the return value, or the invoked object.

You can also configure custom metrics and transaction snapshot data to collect for the backend.

See [Configurations for Custom Exit Points](#) for suggested custom configurations for some common backends.

## To create a custom exit point

1. In the left navigation panel, click **Configure -> Instrumentation**.
2. Click the **Backend Detection** tab.
3. Click the tab corresponding to the backend platform.
4. Select the application or tier for which you are configuring the custom exit point. Backend detection configuration is applied on a hierarchical inheritance model. See [Hierarchical Configuration Model](#).
5. Scroll down to Custom Exit Points.
6. Click **Add** (the + icon).
7. In the Create Custom Exit Point window, click the **Identification** tab if it is not selected.
8. Enter a name for the exit point. This is the name that identifies the backend.
9. Select the type of backend from the Type drop-down menu or check Use Custom if the type is not listed.
10. Configure the class and method name that identify the custom exit point.  
If the method is overloaded, check the Overloaded check box and add the parameters.
11. If you want to restrict metric collection based on a set of criteria that are evaluated at runtime, click **Add Match Condition** and define the match condition(s).  
For example, you may want to collect data only if the value of a specific method parameter contains a certain value.
12. Click **Save**.

The following screenshot shows a custom exit point of Type Cache. This exit point is defined on the `getAll()` method of the specified class. The exit point appears in flow maps as an unresolved backend named CoherenceGetAll.

**Edit Custom Exit Point**

Name:	CoherenceGetAll	Type:	Cache				
<input checked="" type="radio"/> Identification <input type="radio"/> Custom Metrics <input type="radio"/> Snapshot Data							
<b>Define the class and method name which, when called, will be identified as a Custom Exit Point</b>							
Class	that implements an Interface which	equals	com.tangosol.net.NamedCache				
Method Name	getAll	<input type="checkbox"/> Is this Method Overloaded?					
Method Parameters (optional)							
<input type="button" value="Add Parameter"/>							
Match Conditions (optional)							
<input type="button" value="Add Match Condition"/>							
<b>Calls to the specified class and method name can be further split based by a combination of match conditions.</b>							
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Cachename</td> <td>Collect data from the invoked object and capture the result.</td> </tr> </tbody> </table>		Name	Description	Cachename	Collect data from the invoked object and capture the result.	<input type="button" value="Add"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>	
Name	Description						
Cachename	Collect data from the invoked object and capture the result.						
<input type="button" value="Cancel"/> <input type="button" value="Save"/>							

## To split an exit point

1. Click **Add**.
2. Enter a display name for the split exit point.
3. Specify the source of the data (parameter, return value, or invoked object).
4. Specify the operation to invoke on the source of the data (`toString()` or getter chain for complex objects).
5. Click **Save**.

The following example shows a split configuration of the previously created CoherenceGetAll exit point based on the `getCacheName()` method of the invoked object.

**Edit Custom Exit Point Identifier**

Specify the parameter index or indicate if it the return value of the diagnostic data to be collected.  
Simple getters without parameters can be used on the parameter or the return value to be displayed against the display name specified here.

Display Name

*Create your own name for the data collected. This will be the display name for the data in Request Snapshots*

Collect Data From  Method Parameter @ Index:

Return Value

Invoked Object

Operation on Invoked Object  Use `toString()`

Use Getter Chain

*for example: `getAccount().getBalance()`*

## To group an exit point

You can group methods as a single exit point. The only requirement is that these methods point to the same key.

For example, ACME Online has an exit point for `NamedCache.getAll`. This exit point has a split configuration of `getCacheName()` on the invoked object as illustrated in the previous screenshot.

Suppose we also define another exit point for `NamedCache.entrySet`. This is another exit point, but it has the split configuration that has `getCacheName()` method of the invoked object.

**Edit Custom Exit Point**

Name:	CoherenceCacheAccess	Type:	Cache				
<input checked="" type="radio"/> Identification <input type="radio"/> Custom Metrics <input type="radio"/> Snapshot Data							
<b>Define the class and method name which, when called, will be identified as a Custom Exit Point:</b>							
Class	that implements an Interface which		equals com.tangosol.net.NamedCache				
Method Name	entrySet	<input type="checkbox"/> Is this Method Overloaded?					
Method Parameters (optional)							
<b>Add Parameter</b>							
Match Conditions (optional)							
<b>Add Match Condition</b>							
<b>Calls to the specified class and method name can be further split based by a combination of method parameters and match conditions.</b>							
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>CacheName</td> <td>Collect data from the invoked object and capture the result of getCacheName().</td> </tr> </tbody> </table>				Name	Description	CacheName	Collect data from the invoked object and capture the result of getCacheName().
Name	Description						
CacheName	Collect data from the invoked object and capture the result of getCacheName().						
<input type="button" value="Add"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>							

If the getAll() and the entrySet() methods point to the same cache name, they will point to the same backend.

Matching name-value pairs identify the back-end. In this case, there is only one key, i.e. cache name, has to be matched. So, here both exit points have the same name for the cache and they resolve to the same backend.

## To define custom metrics for a custom exit point

Custom metrics are collected in addition to the standard metrics.

The result of the data collected from the method invocation must be an integer value, which will either be averaged or added per minute, depending on your selection of data roll-up.

To configure custom business metrics that can be generated from the Java method invocation:

1. Click the **Custom Metrics** tab.
2. Click **Add**.
3. In the Add Custom Metric window, enter a name for the metric.
4. Select the Collect Data From radio button to specify the source of the metric data.
5. Select the Operation on Method Parameter to specify how the metric data is processed.
6. Select how the data should be rolled up (average or sum) from the Data Rollup drop-down menu.
7. Click **Create Custom Metric**.

## To define transaction snapshot data collected

1. Click the **Snapshot Data** tab.
2. Click **Add**.
3. In the Add Snapshot Data window, enter a display name for the snapshot data.
4. Select the Collect Data From radio button to specify the source of the snapshot data.
5. Select the Operation on Method Parameter to specify how the snapshot data is processed.
5. Click **Save**.

## Learn More

- [Configurations for Custom Exit Points](#)

# Configure Stale Backend Removal

- [Stale or Orphaned Backends](#)
  - To configure automatic stale backend deletion
- [Learn More](#)

## Stale or Orphaned Backends

A stale backend (also called an orphaned backend) is an unresolved backend for which AppDynamics has previously captured metrics, but which has experienced no metric activity for the configured time period.

You can configure AppDynamics to remove stale backends (database and remote service servers) automatically at a regular interval by setting the Controller-level global `backend.permanent.deletion.period` property.

If you are sharing a Controller on a SaaS account, contact [AppDynamics Support](#).

When this global property is enabled, AppDynamics removes all the stale backends in the managed environment, including any metrics previously collected for them and any health rules referencing their metrics. After a backend is removed, its metrics no longer appear in the Metric Browser and health rules that reference those metrics do not fire. You should remove any health rules conditions that reference metrics in stale backends.

By default, automatic removal of stale backends is enabled with a default interval of one month. The beginning of the interval is the last time that activity was reported on the backend (that `Calls per Minute > 0`) and the end of the interval is the time at which the backend is deleted.

You can modify the interval. This may be advisable, especially for large installations, since the maximum number of backends removed in one pass is 50. The minimum interval is one week.

You can also disable automatic removal by setting the interval to 0.

The automatic backend removal is logged in the `server.log` with the message:

```
BACKEND PURGER deleting unresolved stale backend ids:
```

followed by a list of the IDs being deleted.

## To configure automatic stale backend deletion

1. Log into the Controller Admin console using the admin account.

<http://<controller-installer-host>:<port>/controller/admin.html>

2. Select Controller Settings.
3. Scroll down to the backend.permanent.deletion.period property.
4. Enter the new interval in hours.
5. Click **Save**.

## Learn More

- Backend Monitoring

# Configure Policies

- Structure of the Policy Wizard
  - To access the Policy Wizard
- Configuring the Policy Trigger
  - To configure policy triggers
- Configuring the Policy Actions
  - To configure policy actions
- Learn More

## Structure of the Policy Wizard

The Policy Wizard contains two panels:

- Trigger: Sets the policy name, enabled status, events that trigger the policy, entities that are affected by the policy
- Actions: Sets the actions to take when the policy is triggered.

### To access the Policy Wizard

1. Click **Alert & Respond -> Policies** in the left navigation pane.
2. Do one of the following:
  - To edit an existing policy, select the policy and click the edit icon.
  - To remove an existing policy, select the policy and click the delete icon.
  - To create a new policy, click the "+" icon.

## Configuring the Policy Trigger

The policy trigger panel defines the events and objects generating the events that cause the policy to fire and invoke its actions.

For policy triggers that depend on health violation events, the health rules must be created before you can create a policy on them. See [Health Rules](#) and [Troubleshoot Health Rule Violations](#).

### To configure policy triggers

1. Open the Policy Wizard for creating a new policy or for editing an existing one. See [To access the Policy Wizard](#).
2. Enter a name for the policy in the Name field.
3. To enable the policy, check the Enabled check box. To disable the policy, clear the Enabled check box.
4. On the left, click **Trigger** if it is not already selected.
5. Check the check boxes for the events that will trigger the policy. You may need to click the down arrow to expose specific events

within an event category.

If you select any health rule violation events, you can choose whether any (all) health rule violations or only specific health rule violations will trigger the policy.

To designate specific health rule violations, select These Health Rules, click the "+" icon, and then choose the health rules from the embedded health rule browser.

This Policy will fire when ▼ any of these Events occur on ▶ any object

#### Health Rule Violation Events

- Health Rule Violation Started - Warning
- Health Rule Violation Started - Critical
- Health Rule Violation Upgraded - Warning to Critical
- Health Rule Violation Downgraded - Critical to Warning
- Health Rule Violation Ended

#### What Health Rules?

- Any Health Rule
- These Health Rules:

 Business Transaction response time is much higher than normal

+ Select Health Rule      1 of 12 Selected      Create New Health Rule

#### Other Events

- ▼  Slow Transactions
- Slow Transactions
- Very Slow Transactions
- Stalled Transactions
- ▼  Code Problems
- Code Deadlock
- Resource Pool Limit Reached
- ▼  Application Changes
- Application Deployment
- App Server Restart
- Application Configuration Change
- ▶  AppDynamics Config Warnings

6. When you have finished selecting the events that trigger the policy, click "any object" to configure the objects of events that will trigger the policy.

This Policy will fire when ▼ any of these Events occur on ▶ any object



If you select **Any Objects** the policy will be triggered by the configured events when they occur on any object in your application. To restrict the policy to specific objects, select **Any of these specified objects** and then choose the objects from the embedded object browser.

For example, the following policy fires only on the specified events generated on Node\_8002 and Node\_8003. You can similarly restrict the objects to specific tiers, business transactions or exceptions.

Create Policy

Name: Node8002\_Node8003Policy Enabled:

TRIGGER This Policy will fire when ▼ any of these Events occur on ▶ these specified objects

ACTIONS

Any objects  
 Any of these specified objects:  
 Business Transactions  
 Tiers and Nodes  
 Exceptions

Events that are associated to ANY of the specified objects will make this Policy fire

Select Tiers or Nodes  
 Tiers  
 Nodes

Select Nodes  
Type of nodes: Java and .NET nodes  
These specified Nodes: ▾

Selected Nodes (2)

Name	Tier
Node_8002	Inventory Server
Node_8003	ECommerce Server

< ADD

Other Nodes (2)

Name	Tier
Node_8001	Order Processing Ser
Node_8000	ECommerce Server

7. Click **Save** to save the policy configuration.

## Configuring the Policy Actions

The policy actions panel defines the actions that the policy automatically initiates when the trigger fires.

The actions must be created before you can create a policy that fires them. See [Actions](#) and the documentation for individual types of actions (notification actions, remediation actions, etc.) for information on creating an action.

## To configure policy actions

1. If you have not already done so, open the policy wizard and edit the policy to which you want to add actions. See [To access the Policy Wizard](#).

2. On the left, click **Actions** if it is not already selected.

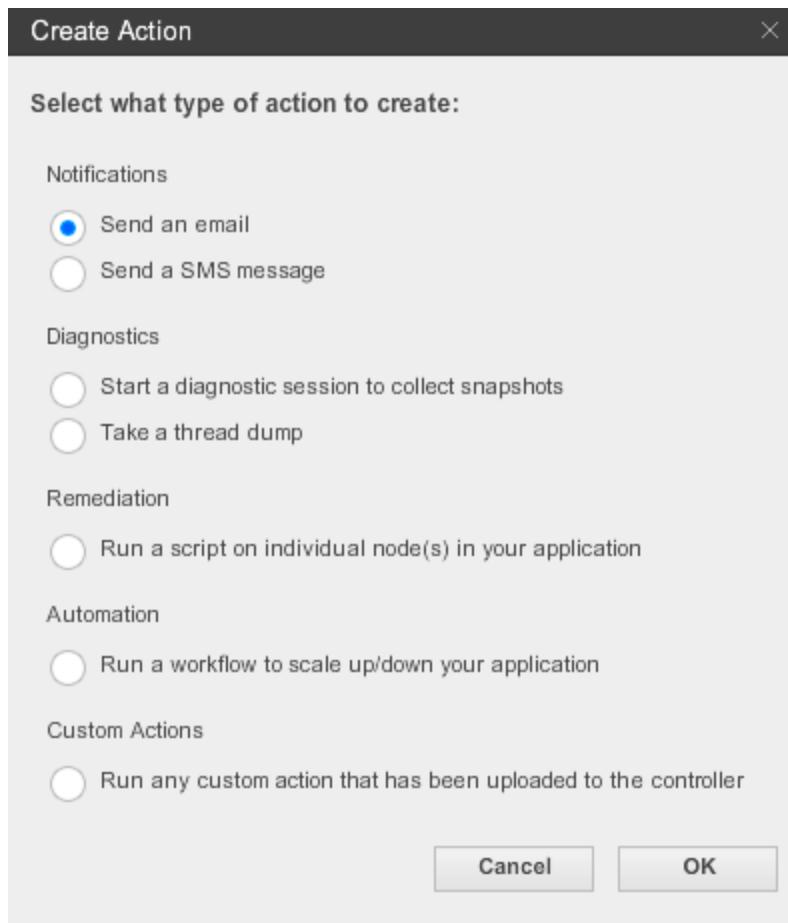
3. Click "+" icon and then **Select Action**.

The list of actions appears. You can filter the list by checking the check boxes for the types of actions you want to see.

4. In the list of actions, select the action that you want this policy to fire and click **Select**.

Name	Approval?	Type
ThreadDump	Yes	Thread dump
IncreasePool	Yes	Run local script

If you do not see an appropriate action for your needs, click **Create Action** to create an action. See [Actions](#).



Then select the newly-created action to assign it to the policy that you are configuring.

5. Click **Save** in the Policy Wizard.

## Learn More

- Policies
- Events
- Health Rules
- Troubleshoot Health Rule Violations
- Actions
- Notification Actions
- Diagnostic Actions
- Remediation Actions (Java only)
- Auto-Scaling Actions
- Custom Actions
- Create a Workflow
- Workflow Automation

## Configure Health Rules

- Structure of the Health Rule Wizard
- Configuring Generic Health Rule Settings
  - To Configure Generic Health Rule Settings
  - To Create a New Health Rule Schedule
- Configuring Affected Entities

- To Configure Affected Entities
- To Configure Affected Entities for Custom Health Rule Types
- Configuring Health Rule Conditions
  - To Configure a Condition
  - To Configure a Condition Component
  - To build an expression
- Configuring the Evaluation Scope

This topic describes the detailed steps for configuring health rules using the Health Rule Wizard.

## Structure of the Health Rule Wizard

To access the Health Rule Wizard:

1. Click **Alert & Respond -> Health Rules**.
2. To edit an existing health rule, in the left panel of the health rule list, select the health rule and click the Edit icon.
3. To remove an existing health rule click the delete icon.
4. To create a new health rule, click "+" .

The Health Rule Wizard opens. It contains four panels:

1. **Overview:** Sets the health rule name, enabled status, health rule type, health rule enabled period, health rule evaluation time
2. **Affects:** Sets the entities evaluated by the health rule. The options presented vary according to the health rule type set in the Overview panel.
3. **Critical Condition:** Sets the conditions, whether all or any of the conditions constitutes a health rule violation, the evaluation scope (BT and node health policies defined at the tier level only); includes an expression builder to create complex expressions containing multiple metrics.
4. **Warning Condition:** Settings are identical to Critical Condition, but configured separately.

You can navigate among these panels using the **Back** and **Next** buttons at the bottom of each panel or by clicking their entries in the left panel of the wizard. You should configure the panels in order, because the configuration of the health rule type in the Overview panel determines the available affected entities in the Affects panel as well as the available metrics in the Condition panels.

## Configuring Generic Heath Rule Settings

You configure generic settings for all health rules in the Overview panel.

### To Configure Generic Health Rule Settings

1. In the Overview panel of the Health Rule Wizard, if you are creating a new health rule, enter the health rule name in the Name field. If you are editing an existing health rule, the name will already be there. You can change the name of the health rule in the Name field.
  2. To enable the health rule check the Enabled check box. To disable the health rule, clear the Enabled check box.
  3. Click one of the health rule types in the health rule type list to select the health rule type. This setting affects metrics offered for configuration in subsequent windows in the wizard, so you must select a health rule type before continuing to other windows. If none of the predefined health rules are applicable for your needs, select Custom. Custom health rule types are offered all metrics for subsequent configuration in the Condition panels.
  4. If the health rule is always (24/7) enabled, check the Always check box. If the health rule is enabled only at certain times, clear the Always check box and either:
    - Select a predefined time interval from the drop-down menu.
- or
- Click **Create New Schedule**. The Create Schedule window opens. See [To Create a New Health Rule Schedule](#).

The time is the time at the site of the controller, not the app agent. For example, if the enabled time is set to 5pm-6pm, Mon-Fri and the controller is in San Francisco but the app agent is in Dubai, the health rule engine uses San Francisco time.

5. Click the dropdown menu and select a value between 1 and 360 minutes for the evaluation window. This is the amount of recent data to use to determine whether a health rule violation exists. This value applies to both critical and warning conditions. See [Health Rule Evaluation Window](#).

6. In the Wait Time after Violation section, enter the number of minutes to wait before evaluating the rule again for the same affected entity in which the violation occurred. See [Health Rule Wait Time After Violation](#).

7. Click **Save**.

8. Click **Next**.

## To Create a New Health Rule Schedule

1. In the Overview window of the Health Wizard, clear the Always check box if it is checked.

2. Click **Create New Schedule**.

2. Enter a name for the schedule.

3. Enter an optional description of the schedule.

3. Enter the start and end times for the schedule as cron expressions. See <http://www.quartz-scheduler.org/documentation/quartz-2.1.x/tutorials/crontrigger> for details about and examples of cron syntax.

5. Click **Save**.

## Configuring Affected Entities

This feature lets you define health rules broadly or fine-tune them very precisely to affect specific entities in your application.

The choices offered for configuring affected entities vary according to the health rule type previously configured in the Overview panel.

## To Configure Affected Entities

1. In the Affects panel, select the entities affected by this health rule from the drop-down menu.

The entity affected and the choices presented in the menu depend on the health rule type configured in the Overview window.

See [Entities Affected by a Health Rule](#) for information about the types of entities that can be affected by the various health rule types.

If you have selected nodes based on matching criteria, specify the matching criteria. Nodes can be matched based on the node name, meta-information properties, environment variables, and JVM system environment properties (Java only).

If you are configuring a JMX health rule, select the JMX objects that the health rule is evaluated on.

If the configured health rule type is custom, see [To Configure Affected Entities for Custom Health Rule Types](#).

2. Click **Save**.

3. Click **Next**.

## To Configure Affected Entities for Custom Health Rule Types

1. In the Affects window of the Health Rule Wizard, select the entity level that the health rule affects: Business Transaction Performance, Node Performance, or Application Performance.

2. If the health rule affects Business Transaction Performance, select the Business Transaction performance radio button and the Business Transaction affected from the drop-down menu. You can use the search field in the Select a Business Transaction browser that opens to locate the specific Business Transaction.

3. If the health rule affects Node Performance:

- a. Select the Node Performance radio button.
- b. Click **Node (for w new configuration) or \*Change** (to modify an existing configuration).
- c. In the Node browser that opens, navigate to the node which you want the health rule to affect.

- d. Click **Select**.
3. If the health rule is not specific to a Business Transaction or Node, select Application Performance.

## Configuring Health Rule Conditions

The configuration processes for critical and warning conditions are identical.

The Health Rule Wizard provides the ability to copy settings between Critical and Warning condition panels to facilitate the creation of similar health rules on the same metrics, possibly with different thresholds, or to assign different alerts or actions. For example, if you have already defined a critical condition and you want to create a warning condition that is similar, in the Warning Condition window click **Copy from Critical Condition** to populate the fields with settings from the Critical condition.

The high-level process for configuring conditions is:

1. Determine how many metrics and which metrics the health rule will evaluate. For each metric for which you want to use to evaluate performance, create a condition. You can use a single metric or build a mathematical expression based on multiple metrics and operations.
2. Decide whether the health rule is violated if all of the tests are true or if any single test is true.
3. For business transaction performance health rules and node health rule types that specify affected entities at the tier level, decide how many of the nodes must be violating the health rule to constitute a violation. See [Health Rule Evaluation Scope](#).
4. To configure a critical condition use the Critical Condition window. To configure a warning condition use the Warning Condition window.

### To Configure a Condition

1. In the Conditional Condition or Warning Condition window, click **+ Add Condition** to add a new condition component. The row defining the component opens. See [To Configure a Condition Component](#). Continue to add components to the condition as needed.
2. From the drop-down menu above the components, select All if all of the components must evaluate to true to constitute violation of the rule. Select Any if a health rule violation exists if any single component is true.
3. For Business Transaction and node health rules that define the affected entities by tier rather than node, configure the evaluation scope. See [Health Rule Evaluation Scope](#).

Health Rule will violate if the conditions above evaluate to true for:

- the Tier Average (the aggregate of all Nodes)
- Any Node
- % of the Nodes
- of the Nodes

### To Configure a Condition Component

1. In the first field of the condition row, name the condition. This name is used in the generated notification text and in the AppDynamics console to identify the violation.

2. To select the metric on which the condition is based, do one of the following:

- To specify a simple metric, click the metric icon to open a small metric browser. The browser displays metrics appropriate to the health rule type. Select the metric to monitor and click **Select Metric**. The selected metric appears in the test configuration.

or

- To build an expression using multiple metrics, click the gear icon at the end of the row and select **Use a mathematical expression of 2 or more metric values.**  
This opens the mathematical expression builder where you can construct the expression to use as the metric. See [To Build an Expression](#).

4. From the Value drop-down menu before the metric, select the qualifier to apply to the metric from the following options:

- Minimum
- Maximum
- Value
- Sum
- Count
- Current

Value refers to a metric over the entire evaluation time length configured in the Overview window. Current is the latest value.

5. From the drop-down menu after the metric, select the value against which the metric is evaluated.

- To compare the metric with a literal value, select < **Specific value** or > **Specific Value** from the menu, then enter the specific value in the text field. For example:

```
Value of Errors per Minute > 100
```

- To compare the metric with a baseline, select < **Baseline** or > **Baseline** from the drop-down menu, and then select the baseline to use, the numeric qualifier of the unit of evaluation and the unit of evaluation. For example:

```
Maximum of Average Response Time is > Baseline of the Daily Trend by 3 X the Baseline Standard Deviation
```

See [Baselines and Periodic Trends](#) for information about the baseline options.

6. Click **Save**.

You can remove a component from the condition by clicking the delete icon.

## To build an expression

To access the expression builder to create a complex expression as the basis of a condition, click the gear icon at the end of the row and select **Use a mathematical expression of 2 or more metric values.**

In the expression builder, use the Expression pane to construct the expression.

Use the Variable Declaration pane to define variables based on metrics to use in the expression.

1. In Variable Declaration pane of the Mathematical Expression builder, click **+ Add variable** to add a variable.
2. In the Variable Name field enter a name for the variable.
3. Click **Select a metric** to open a small metric browser  
Follow the instructions in [To configure a test](#) to select a metric.
4. From the drop-down menu select the qualifier for the metric.
5. Repeat steps 1 through 4 for each metric that you will use in the expression.  
You can remove a variable by clicking the delete icon.
6. Build the expression by typing the expression in the Expression pane. Click the **Insert Variable** button to insert variables created in the Variable Declaration pane.

**Mathematical Expression**

**Expression**

Type in a mathematical expression below using a combination of mathematical operators (eg. +,-,\*,/), and ()  
constants (eg. 1,2, etc.), and variables declared above enclosed in curly brackets (eg. {numSlows}). An example of a valid Expression would be  $2+\{x\}$  or  $\{variableName1\} + (4 * (5 - \{variableName2\}))$ .

{numSlows} + {numVerySlows} +

**Variable Declaration**

First declare variables to represent your desired Metric Expressions. For example, you can  
= Value of Number of Slow Calls.

Variable Name	Variable Definition
numStalls	Value of Stall Count
numVerySlow	Value of Number of Slow Calls
numSlows	Value of Number of Slow Calls

**Insert Variable...**

numStalls - Value of Stall Count  
numVerySlow - Value of Number of Slow Calls  
numSlows - Value of Number of Slow Calls

**Add Variable**

**Cancel** **Use Expression** **Back** **Next >**

7. When the expression is built, click **Use Expression**.  
The expression appears as the metric in the condition configuration window.

condition 1

{numSlows} + {numVerySlows} + {numStalls} [Edit Expression](#)

is

## Configuring the Evaluation Scope

This setting is applicable to business transaction performance type policies and node health policies that specify the affected entities at the tier level.

If the **Health Rule will violate if the conditions above evaluate to true** section is visible, click the appropriate radio button to set the evaluation scope.

If you select percentage of nodes, enter the percentage. If you select number of nodes, enter the absolute number of nodes.

## Configure Email Digests

- Structure of the Email Digest Wizard

- Configuring the Email Digest
  - To configure content settings
  - To configure digest recipients
  - To configure the digest interval
- Learn More

## Structure of the Email Digest Wizard

To access the Email Digest Wizard:

1. Click **Alert & Respond -> Email Digests**.
2. To edit an existing email digest, select the digest and click the Edit icon.
3. To remove an existing digest click the delete icon.
4. To create a new digest, click "+" .

The Email Digest Wizard opens. It contains four panels:

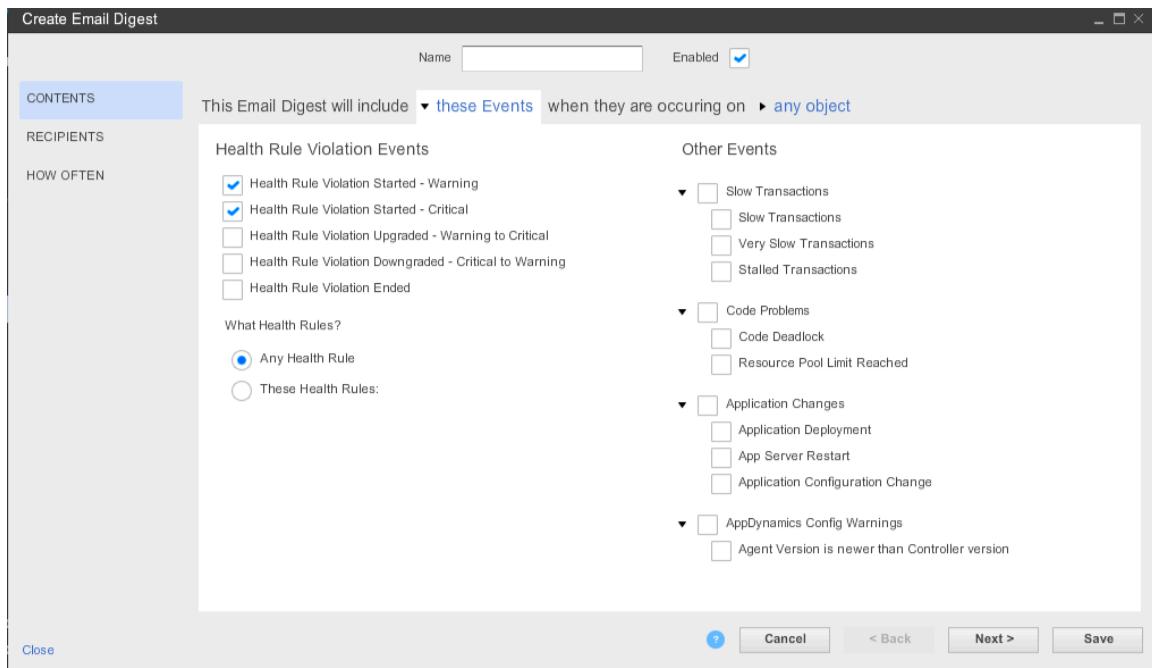
1. Contents: Sets the digest name, enabled status, health rule type, events and objects that trigger the sending of the digest
2. Recipients Adds the email addresses of the digest recipients.
4. How Often: Sets how often the digest is sent, in hours.

You can navigate among these panels using the Back and Next buttons at the bottom of each panel or by clicking their entries in the left panel of the wizard.

## Configuring the Email Digest

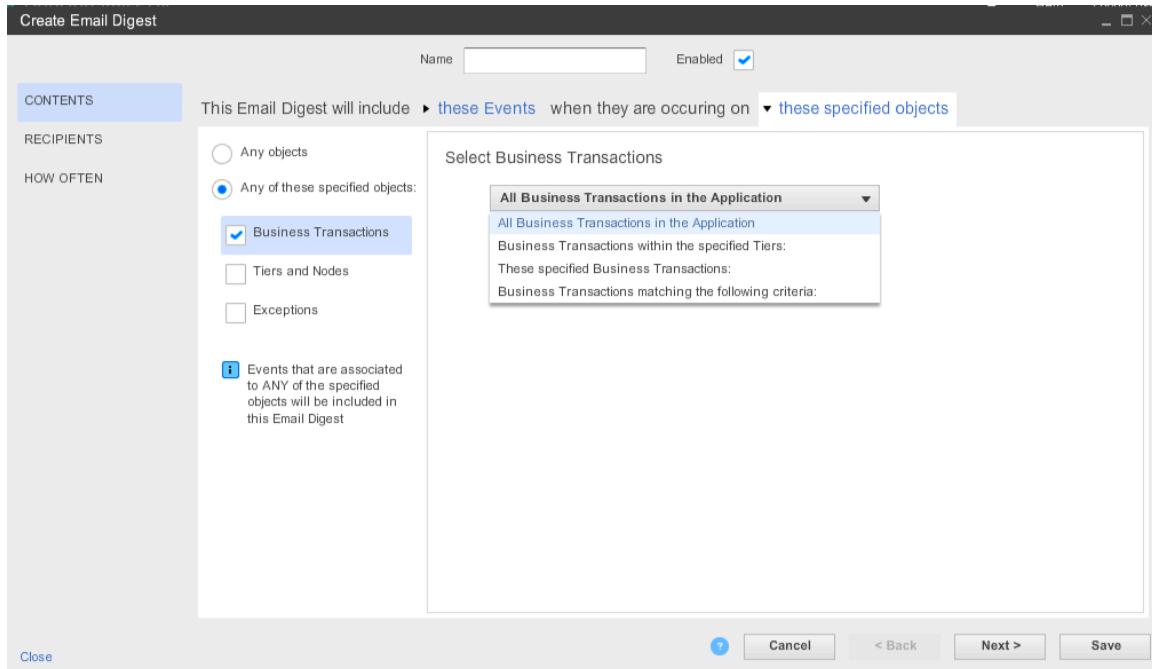
### To configure content settings

1. In the Content panel of the Email Digest Wizard, if you are creating a new digest, enter the digest name in the Name field. If you are editing an existing digest, the name will already be there. You can change the name of the digest in the Name field.
2. To enable sending the digest check the Enabled check box. To disable the digest, clear the Enabled check box.
3. Check the check boxes for the events that will be included in the digest. You may need to click the down arrow to expose specific events within an event category.



4. When you have finished selecting the events in the digest, you can click "any object" to refine the contents by specifying only events that affect certain objects in the application. If you select Any Objects the digest will include configured events when they occur on any object in your application.

To restrict the policy to specific objects, select Any of these specified objects and then choose the objects from the embedded object browser.



5. Click **Save** to save the digest configuration.

## To configure digest recipients

Configuration of digest recipients involves selecting or creating an email notification action for every recipient of the digest. You can create these notification actions from this Email Digest Wizard as well as from the **Actions** menu.

1. In the Recipient panel of the Email Digest Wizard, if you are creating a new digest, enter the digest name in the Name field. If you are modifying an existing digest, double-click the digest in the list.
  2. To edit an existing recipient, select the recipient from the list and double-click or click the Edit icon.
  3. To remove an existing recipient from the email digest, select the recipient from the list and click the delete icon.
  4. To add a recipient, click the "+" sign.
  5. Do one of the following:
    - To add a recipient who has already been configured (i.e. an existing email notification action), click **Select Action**, select the email notification from the list, and click **Select**.
- or
- Click **Create Email Action**, enter the email address of the recipient in the text field.
6. In the Add Action screen, you can also add an optional note to include in the email.
  7. Click **Save** to add the recipient.
  8. Click **Save** in the Email Digest Wizard to save the digest configuration.

## To configure the digest interval

1. Click How Often to access the How Often panel.
- 2 In the text field enter an integer to indicate the number of hours between digests.
3. Click **Save** to save the digest configuration.

## Learn More

- [Email Digests](#)
- [Policies](#)
- [Health Rules](#)
- [Notification Actions](#)

# Configure Call Graphs

- [Call Graph Settings](#)
  - To access call graph configuration screens
  - [Call Graph Granularity](#)
  - [Packages or Namespaces to Exclude from Call Graphs](#)
    - To exclude specific packages or namespaces from call graphs
    - To Include Specific Sub-packages (Sub-namespaces) or Classes from the Excluded Packages
  - [SQL Capture Settings](#)
    - To Configure SQL Bind Variables
  - [Slow Transaction Snapshot Collection \(Java only\)](#)
    - To Enable / Disable Aggressive Slow Snapshot Collection
  - [Learn More](#)

This topic describes how to configure call graphs.

## Call Graph Settings

The Call Graph Settings window lets you configure thresholds that affect performance, which packages or namespaces to include in call graphs, and how much detail about SQL statements to capture.

## To access call graph configuration screens

1. In the left navigation pane, click **Configure -> Instrumentation**.
2. Click the **Call Graph Settings** tab.
3. Click the **Java Call Graph Settings** or **.NET Call Graph Settings** tab depending on your framework.

Whenever you create or modify a call graph setting in these screens, click the **Save Call Graph Settings** button to save your configuration.

## Call Graph Granularity

You can control the granularity for call-graphs using following settings:

- **Control granularity for Methods:** To ensure low performance overhead, choose a threshold in milliseconds for method execution time. Methods taking less than the time specified here will be filtered out of the call graphs.
- **Control granularity for SQL calls:** You can also specify a threshold for SQL queries. SQL queries taking more than the specified time in milliseconds will be filtered out of the call-graphs. Also see [App Agent for Java Performance Tuning](#).

## Packages or Namespaces to Exclude from Call Graphs

A call graph can potentially contain hundreds of methods. You can exclude packages (Java) or namespaces (.NET) with classes that you do not want to monitor.

For Java, some packages are excluded by default. These are visible in the Excluded Packages list. The packages that are excluded by default cannot be removed. However, you can include a particular sub-package from an excluded package. See [To Include Specific Sub-packages \(Sub-namespaces\) or Classes from the Excluded Packages](#).

### To exclude specific packages or namespaces from call graphs

1. Click **Add Custom Package Exclude** (Java) or **Add Custom Namespace Exclude** (.NET).
2. Enter the name and description of the package or namespace to exclude.
3. Click **Add**.

### To Include Specific Sub-packages (Sub-namespaces) or Classes from the Excluded Packages

1. Click **Add Always Show Package/Class** (Java) or **Add Always Show Namespace/Class** (.NET).
2. Specify the subpackage/class or namespace/class and a description to include in the call graph at all times.
3. Click **Add**.

By default, AppDynamics provides support for identifying methods for Jersey REST framework and Apache EJB classes in the call graph.

## SQL Capture Settings

Often the SQL Calls section does not display the raw values in a SQL query, as shown in the following query:

```
INSERT INTO ORDERREQUEST ( ITEM_ID, NOTES ) VALUES ( ?, ? )
```

Replacing the literals in a query with parameter markers in this way is called normalizing the query.

Normalizing a query prevents display of sensitive data, such as social security numbers or credit card numbers, which are potential query parameters. Normalizing queries also helps to organize SQL data by flattening the parameter values for the query so that the statistics from different executions of the query can be aggregated and compared against one another.

However, during troubleshooting, you may want to display the values of the bind variables.

## To Configure SQL Bind Variables

1. In the **Call Graph Settings** tab, scroll down to the SQL Capture Settings section.
2. Select one of the following:

- **Capture Raw SQL:** Select this option to see raw SQL data (this captures raw SQL data along with the parameter values). Raw SQL data includes prepared statement bind variables or raw statements. By default, the private SQL data and queries that take less than 10 ms are not captured.
- **Filter Parameter values:** Select this option to filter certain parameter values from the captured SQL.

SQL Capture Settings

Capture Raw SQL - This will capture raw SQL statements with parameter values included. For example "select \* from user where user\_SSN = '111-11-1111';"  
 Filter Parameter Values - This will filter out all parameter values from the captured SQL. For example "select \* from user where user\_SSN = '?';"

3. Click **Save Call Graph Settings**.

## Slow Transaction Snapshot Collection (Java only)

Normally AppDynamics captures the full execution path of a request after the business transaction threshold for slow requests has been crossed. Before a request becomes problematic (slow, stalled or error), AppDynamics captures partial call graphs that do not show the invocation stack from before the request started to slow.

You can configure more aggressive snapshot collection to capture the full execution path of requests before thresholds are crossed. This feature is currently supported only for Java frameworks.

By default aggressive snapshot collection is disabled to minimize overhead. You may want to enable it when you are starting to experience performance problems for which you want to find root cause.

This configuration affects snapshot collection at the application level. You can also enable and disable this feature at the node level using the enable-hotspot-snapshots node property. For information about setting node properties see [App Agent Node Properties](#).

## To Enable / Disable Aggressive Slow Snapshot Collection

1. Select the application for which you want to enable or disable aggressive snapshot collection.
2. In the left navigation panel click **Configure -> Instrumentation**.
3. Click the **Call Graph Settings** tab, and then the **Java Call Graph Settings** sub-tab.
4. At the bottom of the Call Graph Settings screen, in the Slow Transaction Snapshot Collection section, check the **Enable Aggressive Slow Snapshot Collection** checkbox to enable aggressive collection. Clear the checkbox to disable aggressive collection.
5. Click **Save Call Graph Settings**.

## Learn More

- [Call Graphs](#)
- [App Agent for Java Performance Tuning](#)
- [Transaction Snapshots](#)

## Configure Business Metric Information Points

- [To create a business metric information point](#)
- [Learn More](#)

AppDynamics provides information points to specify code and business metrics. See [Code Metrics and Business Metrics](#).

## To create a business metric information point

Configuring business metrics, also called custom metrics, involves:

- Identify the parameter or the return type of a method invocation.

- Derive a value from the method invocation to represent the metric.
- Choose either to generate the sum of the value over a period of time or to average it.

1. Select your application.
2. Click **Analyze -> Information Points**.
3. Click **Add** (the + symbol).
4. Enter a name for the information point.
5. Select the agent type.
6. Define the class and the method on which the information point is based.

**Edit Information Point - Book Quantity**

Specify Name and Agent Type for this Info Point

Name	Book Quantity
Agent Type	<input checked="" type="radio"/> Java <input type="radio"/> .NET

Define the Method Signature

Specify the Java method signature with the fully qualified class/interface/super-class/annotation name and the method name. If the method is overloaded you will need to enter the parameter types.

Class	with a Class Name that	equals	com.appdynamics.inventory.item
Method Name	getQuantity	<input type="checkbox"/> Is this Method Overloaded?	

Method Parameters (optional)

Add Parameter

Match Conditions (optional)

Add Match Condition

Define Custom Metrics using the Method Parameters or Return Value

Name	Custom Metric
Book Quantity	SUM (ReturnValue.toString())

Add Edit Delete

Cancel Save

7. (Optional) If the method is overloaded, check the **Is this Method Overloaded** option and add the parameters.
8. (Optional) Define match conditions. A match condition ensures that only those invocations that match the configured conditions for the Information Point will be captured.
9. In the Define Custom Metrics section, click **Add**. Choose the parameter or return value from which the metric needs to be generated.
10. Choose the data rollup over time.
11. Click **Save**.
12.  Important: Restart the application server on which this rule will be applied.

For example, this definition counts the sum of books ordered in the Acme Bookstore application.

**Edit Information Point - Book Quantity**

Specify Name and Agent Type for this Info Point

Name: Book Quantity  
Agent Type: Java (.NET)

Define the Method Signature

Specify the Java method signature with the fully qualified class/interface/super-class/annotation name and the method name. If the method is overloaded you will need to enter the parameter types.

Class: with a Class Name that equals com.appdynamics.inventory.item  
Method Name: getQuantity Is this Method Overloaded?

Method Parameters (optional)  
Add Parameter

Match Conditions (optional)  
Add Match Condition

Define Custom Metrics using the Method Parameters or Return Value

Name	Custom Metric
Book Quantity	SUM (ReturnValue.toString())

Add Edit Delete Cancel Save

**Information Point Custom Metric Definition**

Custom business metrics can be generated from the Java method invocation using the values of the parameters/return values.

To configure custom metrics, pick the parameter index or return value and an optional getter operation on it if it is a complex object.

Name: Book Quantity

**Data To Collect**

Collect Data From:  Method Parameter @ Index: 0  
 Return Value  
 Invoked Object

Operation on Return Value:  Use `toString()`  
 Use Getter Chain  
 for example: `getAccount().getBalance()`  
[more help](#)

**Data Rollup**

The result of the data collected from the method invocation has to be an integer value which will either be averaged or added per minute, depending on the choice below.

Data Rollup: SUM (Rollup type cannot be changed once created)

Cancel Save

## Learn More

- Business Metrics
- Information Points

# Configure Code Metric Information Points

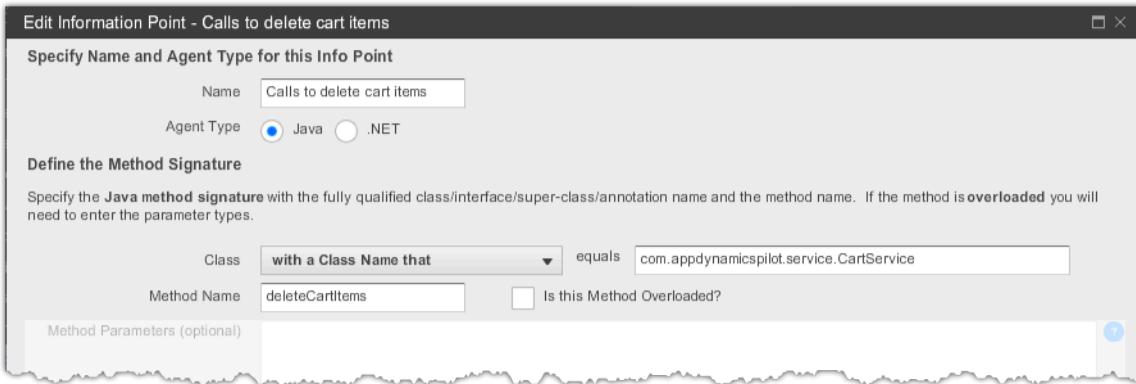
- To create a code metric information point
- Learn More

AppDynamics provides information points to specify code and business metrics. See [Code Metrics](#) and [Business Metrics](#).

## To create a code metric information point

1. Select your application.
  2. Click **Analyze -> Information Points**.
  3. Click **Add** (the + symbol).
  4. Enter a name for the information point.
  5. Select the agent type.
  6. Define the class and the method on which the information point is based.
  7. (Optional) If the method is overloaded, check the **Is this Method Overloaded** option and add the parameters.
8. (Optional) Define match conditions. A match condition ensures that only those invocations that match the configured conditions for the Information Point will be captured. For example, the ACME Bank AccountManager.updateAccount(String customerType, Account newAccountData) method, you can capture only those requests that invoke the parameter "customerType" with value "platinum".

For example, the following configuration counts the number of calls to the method that deletes items from the cart:



## Learn More

- [Code Metrics](#)
- [Information Points](#)

# Configure AppDynamics for Java

# Configure Custom Exit Points (Java)

- Default Backends Discovered by the App Agent for Java
- Configure Custom Exit Points for Java Backends
- To create a custom exit point
  - To split an exit point
  - To group an exit point
- To define custom metrics for a custom exit point
- To define transaction snapshot data collected
- Learn More

AppDynamics provides default automatic discovery for commonly-used backends. If a backend used in your environment is not discovered, first compare the list of default backends to determine whether you need to modify the default configuration. If it is not on the list then configure a custom exit point according to these instructions.

## Default Backends Discovered by the App Agent for Java

The default backends for Java are:

- HTTP
- JDBC
- JMS
- MQ
- RMI
- Thrift (not currently configurable)
- Web Service

To configure a default backend see [Configure Backend Detection](#).

## Configure Custom Exit Points for Java Backends

### Configure Custom Exit Points

Custom exit points provide identification for backend types that are not automatically detected, such as file systems, mainframes etc. For example, you can define a custom exit call to monitor the file system read method. Custom exit points appear as unresolved backends in the flow maps. Unresolved backends are shown on flow maps with this icon



You define a custom exit point by specifying the class and method used to identify the backend. If the method is overloaded, you need to add the parameters to identify the method uniquely.

You can restrict the method invocations for which you want AppDynamics to collect metrics by specifying match conditions for the method. The match conditions can be based on a parameter or the invoked object.

You can also optionally split the exit point based on a method parameter, the return value, or the invoked object.

You can also configure custom metrics and transaction snapshot data to collect for the backend.

See [Configurations for Custom Exit Points](#) for suggested custom configurations for some common backends.

### To create a custom exit point

1. In the left navigation panel, click **Configure -> Instrumentation**.
2. Click the **Backend Detection** tab.
3. Click the tab corresponding to the backend platform.
4. Select the application or tier for which you are configuring the custom exit point. Backend detection configuration is applied on a hierarchical inheritance model. See [Hierarchical Configuration Model](#).
5. Scroll down to Custom Exit Points.

6. Click **Add** (the + icon).
7. In the Create Custom Exit Point window, click the **Identification** tab if it is not selected.
8. Enter a name for the exit point. This is the name that identifies the backend.
9. Select the type of backend from the Type drop-down menu or check Use Custom if the type is not listed.
10. Configure the class and method name that identify the custom exit point.  
If the method is overloaded, check the Overloaded check box and add the parameters.
11. If you want to restrict metric collection based on a set of criteria that are evaluated at runtime, click **Add Match Condition** and define the match condition(s).  
For example, you may want to collect data only if the value of a specific method parameter contains a certain value.
12. Click **Save**.

The following screenshot shows a custom exit point of Type Cache. This exit point is defined on the `getAll()` method of the specified class. The exit point appears in flow maps as an unresolved backend named `CoherenceGetAll`.

**Edit Custom Exit Point**

Name:	CoherenceGetAll	Type:	Cache				
<input checked="" type="radio"/> Identification <input type="radio"/> Custom Metrics <input type="radio"/> Snapshot Data							
<b>Define the class and method name which, when called, will be identified as a Custom Exit Point</b>							
Class	that implements an Interface which	equals	com.tangosol.net.NamedCache				
Method Name	getAll	<input type="checkbox"/> Is this Method Overloaded?					
Method Parameters (optional)							
<b>Add Parameter</b>							
Match Conditions (optional)							
<b>Add Match Condition</b>							
<b>Calls to the specified class and method name can be further split based by a combination of match conditions</b>							
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Cachename</td> <td>Collect data from the invoked object and capture the result of the call.</td> </tr> </tbody> </table>		Name	Description	Cachename	Collect data from the invoked object and capture the result of the call.	<input type="button" value="Add"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>	
Name	Description						
Cachename	Collect data from the invoked object and capture the result of the call.						
<input type="button" value="Cancel"/> <input type="button" value="Save"/>							

## To split an exit point

1. Click **Add**.

2. Enter a display name for the split exit point.
3. Specify the source of the data (parameter, return value, or invoked object).
4. Specify the operation to invoke on the source of the data (`toString()` or getter chain for complex objects).
5. Click **Save**.

The following example shows a split configuration of the previously created CoherenceGetAll exit point based on the `getCacheName()` method of the invoked object.

**Edit Custom Exit Point Identifier**

Specify the parameter index or indicate if it the return value of the diagnostic data to be collected.  
Simple getters without parameters can be used on the parameter or the return value to be displayed against the display name specified here.

Display Name	<input type="text" value="Cachename"/>
<i>Create your own name for the data collected. This will be the display name for the data in Request Snapshots</i>	
Collect Data From	<input type="radio"/> Method Parameter @ Index: <input type="text" value="0"/> <input type="button" value="▲"/> <input type="button" value="▼"/> <input type="radio"/> Return Value
	<input checked="" type="radio"/> Invoked Object
Operation on Invoked Object	<input type="radio"/> Use <code>toString()</code> <input checked="" type="radio"/> Use Getter Chain <input type="text" value="getCacheName()"/> <i>for example: <code>getAccount().getBalance()</code></i>
<input type="button" value="Cancel"/> <input type="button" value="Save"/>	

## To group an exit point

You can group methods as a single exit point. The only requirement is that these methods point to the same key.

For example, ACME Online has an exit point for `NamedCache.getAll`. This exit point has a split configuration of `getCacheName()` on the invoked object as illustrated in the previous screenshot.

Suppose we also define another exit point for `NamedCache.entrySet`. This is another exit point, but it has the split configuration that has `getCacheName()` method of the invoked object.

**Edit Custom Exit Point**

Name:	CoherenceCacheAccess	Type:	Cache				
<input checked="" type="radio"/> Identification <input type="radio"/> Custom Metrics <input type="radio"/> Snapshot Data							
Define the class and method name which, when called, will be identified as a Custom Exit Point:							
Class	that implements an Interface which <input type="button" value="▼"/> equals com.tangosol.net.NamedCache						
Method Name	entrySet	<input type="checkbox"/> Is this Method Overloaded?					
Method Parameters (optional)							
<input type="button" value="Add Parameter"/>							
Match Conditions (optional)							
<input type="button" value="Add Match Condition"/>							
Calls to the specified class and method name can be further split based by a combination of method parameters and results.							
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>CacheName</td> <td>Collect data from the invoked object and capture the result of getCacheName().</td> </tr> </tbody> </table>				Name	Description	CacheName	Collect data from the invoked object and capture the result of getCacheName().
Name	Description						
CacheName	Collect data from the invoked object and capture the result of getCacheName().						
<input type="button" value="Add"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>							

If the getAll() and the entrySet() methods point to the same cache name, they will point to the same backend.

Matching name-value pairs identify the back-end. In this case, there is only one key, i.e. cache name, has to be matched. So, here both exit points have the same name for the cache and they resolve to the same backend.

## To define custom metrics for a custom exit point

Custom metrics are collected in addition to the standard metrics.

The result of the data collected from the method invocation must be an integer value, which will either be averaged or added per minute, depending on your selection of data roll-up.

To configure custom business metrics that can be generated from the Java method invocation:

1. Click the **Custom Metrics** tab.
2. Click **Add**.
3. In the Add Custom Metric window, enter a name for the metric.
4. Select the Collect Data From radio button to specify the source of the metric data.
5. Select the Operation on Method Parameter to specify how the metric data is processed.
6. Select how the data should be rolled up (average or sum) from the Data Rollup drop-down menu.
7. Click **Create Custom Metric**.

## To define transaction snapshot data collected

1. Click the **Snapshot Data** tab.
2. Click **Add**.
3. In the Add Snapshot Data window, enter a display name for the snapshot data.
4. Select the Collect Data From radio button to specify the source of the snapshot data.
5. Select the Operation on Method Parameter to specify how the snapshot data is processed.
5. Click **Save**.

## Learn More

- Configurations for Custom Exit Points

## Configurations for Custom Exit Points

- Configurations for Coherence Exit Points
- Configurations For Memcached Exit Points
- Configurations for DangaMemcached Exit Points
- Configurations for SAP Exit Points
- Configurations for EhCache Exit Points
- Configurations for Mail Exit Points
- Configurations for LDAP Exit Points
- Configurations for MongoDB Exit Points
- Learn More

This topic suggests custom exit point configurations that you can create for specific backends.

### Configurations for Coherence Exit Points

Name of the Exit Point	Type	Method Name	Match Criteria value for the Class	Class/Interface/Superclass/Annotation Name	Splitting Configuration
Coherence.Put	Cache	put	that implements an interface which	com.tangosol.net.NamedCache	getCacheName()
Coherence.PutAll	Cache	putAll	that implements an interface which	com.tangosol.net.NamedCache	getCacheName()
Coherence.EntrySet	Cache	entrySet	that implements an interface which	com.tangosol.net.NamedCache	getCacheName()
Coherence.KeySet	Cache	keySet	that implements an interface which	com.tangosol.net.NamedCache	getCacheName()
Coherence.Get	Cache	get	that implements an interface which	com.tangosol.net.NamedCache	getCacheName()
Coherence.Remove	Cache	remove	that implements an interface which	com.tangosol.net.NamedCache	getCacheName()

### Configurations For Memcached Exit Points

Name of the Exit Point	Type	Method Name	Match Criteria value for the Class	Class/Interface/Superclass/Annotation Name
------------------------	------	-------------	------------------------------------	--

Memcached.Add	Cache	add	With a class name that	net.spy.memcached.MemcachedClient
Memcached.Set	Cache	set	With a class name that	net.spy.memcached.MemcachedClient
Memcached.Replace	Cache	replace	With a class name that	net.spy.memcached.MemcachedClient
Memcached.CompareAndSwap	Cache	cas	With a class name that	net.spy.memcached.MemcachedClient
Memcached.Get	Cache	get	With a class name that	net.spy.memcached.MemcachedClient
Memcached.Remove	Cache	remove	With a class name that	net.spy.memcached.MemcachedClient

## Configurations for DangaMemcached Exit Points

Name of the Exit Point	Type	Method Name	Match Criteria value for the Class	Class/Interface/Superclass/Annotation Name
Memcached.Add	Cache	add	With a class name that	com.danga.MemCached.MemCachedClient
Memcached.Set	Cache	set	With a class name that	com.danga.MemCached.MemCachedClient
Memcached.Replace	Cache	replace	With a class name that	com.danga.MemCached.MemCachedClient
Memcached.Delete	Cache	delete	With a class name that	com.danga.MemCached.MemCachedClient
Memcached.Get	Cache	get	With a class name that	com.danga.MemCached.MemCachedClient
Memcached.GetBulk	Cache	getBulk	With a class name that	com.danga.MemCached.MemCachedClient
Memcached.GetMultiArray	Cache	getMultiArray	With a class name that	com.danga.MemCached.MemCachedClient
Memcached.GetMulti	Cache	getMulti	With a class name that	com.danga.MemCached.MemCachedClient

## Configurations for SAP Exit Points

Name of the Exit Point	Type	Method Name	Match Criteria value for the Class	Class/Interface/Superclass/Annotation Name
SAP.Execute	SAP	execute	With a class name that	com.sap.mw.jco.rfc.MiddlewareRFC\$Client
SAP.Connect	SAP	connect	With a class name that	com.sap.mw.jco.rfc.MiddlewareRFC\$Client
SAP.Disconnect	SAP	disconnect	With a class name that	com.sap.mw.jco.rfc.MiddlewareRFC\$Client
SAP.Reset	SAP	reset	With a class name that	com.sap.mw.jco.rfc.MiddlewareRFC\$Client
SAP.CreateTID	SAP	createTID	With a class name that	com.sap.mw.jco.rfc.MiddlewareRFC\$Client
SAP.ConfirmTID	SAP	confirmTID	With a class name that	com.sap.mw.jco.rfc.MiddlewareRFC\$Client

## Configurations for EhCache Exit Points

Name of the Exit Point	Type	Method Name	Match Criteria value for the Class	Class/Interface/Superclass/Annotation Name	Splitting Configuration
EhCache.Get	Cache	get	With a class name that	net.sf.ehcache.Cache	getName()
EhCache.Put	Cache	put	With a class name that	net.sf.ehcache.Cache	getName()
EhCache.PutIfAbsent	Cache	putIfAbsent	With a class name that	net.sf.ehcache.Cache	getName()
EhCache.PutQuiet	Cache	putQuiet	With a class name that	net.sf.ehcache.Cache	getName()
EhCache.Remove	Cache	remove	With a class name that	net.sf.ehcache.Cache	getName()
EhCache.RemoveAll	Cache	removeAll	With a class name that	net.sf.ehcache.Cache	getName()

EhCache.RemoveQuiet	Cache	removeQuiet	With a class name that	net.sf.ehcache.Cache	getName()
EhCache.Replace	Cache	replace	With a class name that	net.sf.ehcache.Cache	getName()

## Configurations for Mail Exit Points

Name of the Exit Point	Type	Method Name	Match Criteria value for the Class	Class/Interface/ Superclass/Annotation Name
MailExitPoint.Send	Mail Server	send	With a class name that	javax.mail.Transport
MailExitPoint.SendMessage	Mail Server	sendMessage	With a class name that	javax.mail.Transport

## Configurations for LDAP Exit Points

Name of the Exit Point	Type	Method Name	Match Criteria value for the Class	Class/Interface/ Superclass/Annotation Name
LDAPExitPoint.Bind	LDAP	bind	With a class name that	javax.naming.directory.InitialDirContext
LDAPExitPoint.Rebind	LDAP	rebind	With a class name that	javax.naming.directory.InitialDirContext
LDAPExitPoint.Search	LDAP	search	With a class name that	javax.naming.directory.InitialDirContext
LDAPExitPoint.ModifyAttributes	LDAP	modifyAttributes	With a class name that	javax.naming.directory.InitialDirContext
LDAPExitPoint.GetNextBatch	LDAP	getNextBatch	With a class name that	com.sun.jndi.ldap.LdapNamingEnum
LDAPExitPoint.NextAux	LDAP	nextAux	With a class name that	com.sun.jndi.ldap.LdapNamingEnum
LDAPExitPoint.CreatePooledConnection	LDAP	createPooledConnection	With a class name that	com.sun.jndi.ldap.LdapClientFactory
LDAPExitPoint.Search	LDAP	search	With a class name that	com.sun.jndi.ldap.LdapClientFactory
LDAPExitPoint.Modify	LDAP	modify	With a class name that	com.sun.jndi.ldap.LdapClientFactory

## Configurations for MongoDB Exit Points

Name of the Exit Point	Type	Method Name	Match Criteria value for the Class	Class/Interface/Superclass/Annotation Name	Splitting configuration	Snapshot data
MongoDB.Insert	JDBC	insert	With a class name that	com.mongodb.DBCollection	Invoked_Object.getDB(). getName()	Parameter_0.toString()
MongoDB.Find	JDBC	find	With a class name that	com.mongodb.DBCollection	Invoked_Object.getDB(). getName()	Parameter_0.toString()
MongoDB.Update	JDBC	update	With a class name that	com.mongodb.DBCollection	Invoked_Object.getDB(). getName()	Parameter_0.toString()
MongoDB.Remove	JDBC	remove	With a class name that	com.mongodb.DBCollection	Invoked_Object.getDB(). getName()	Parameter_0.toString()
MongoDB.Apply	JDBC	apply	With a class name that	com.mongodb.DBCollection	Invoked_Object.getDB(). getName()	Parameter_0.toString()

## Learn More

- Configure Backend Detection
- Configure Custom Exit Points

## Code Metric Information Points (Java)

- Code Metric Information Points for Java System Classes
  - To instrument a Java system class
  - Learn More

## Code Metric Information Points for Java System Classes

System classes like `java.lang.*` are by default excluded by AppDynamics. To enable instrumentation for a system class, use code metric information points.

The overhead of instrumenting Java system classes is based on the number of calls. AppDynamics recommends that you instrument only a small number of nodes and monitor the performance for these nodes before adding configuring all the nodes in your system.

### To instrument a Java system class

1. Open the `<agent_home>/conf/app-agent-config.xml` file for the node where you want to enable the metric.
2. Add the fully-qualified system class name to the override exclude section in the XML file. For example, to configure the `java.lang.Socket` class connect method, modify following element:

```
<override-system-exclude filter-type="equals" filter-value="java.lang.Socket" />
```

3. Restart those JVMs for which you have modified the XML file.

## Learn More

- [Code Metrics](#)
- [Configure Code Metric Information Points](#)

## Configure JMX Metrics from MBeans

- JMX Metric Rules and Metrics
  - Using the JMX Metric Rules Configuration Panel
    - To create one or more JMX metrics in the JMX Metric Rules panel
  - Using the MBean Browser
    - To create a metric from an MBean attribute in the MBean Browser
- [Learn More](#)

This topic describes how to create persistent JMX metrics from MBean attributes.

For background information about creating JMX metrics see [Monitor JVMs](#) and [Monitor JMX MBeans](#).

## JMX Metric Rules and Metrics

A JMX Metric Rule maps a set of MBean attributes from one or more MBeans into AppDynamics persistent metrics. You configure a metric rule that creates one or more metrics in the AppDynamics system. You may want to create new metrics if the preconfigured metrics do not provide sufficient visibility into the health of your system.

After the MBean attribute is configured to provide a persistent metric in AppDynamics, you can use it to configure health rules. For details see [Health Rules](#).

To view the MBeans that are reporting currently in your managed environment use the [Metric Browser](#).

You can use the [JMX Metrics Rules Panel](#) or the [Using the MBean Browser](#) to create new metrics.

## Using the JMX Metric Rules Configuration Panel

The JMX Metric Rules Panel is the best way to create metrics for multiple attributes based on the same MBean or for complex matching patterns.

### To create one or more JMX metrics in the JMX Metric Rules panel

1. In the left navigation pane, click **Configure -> Instrumentation**.
2. Click the **JMX** tab.
3. In the JMX Metric Configurations pane, click the Java platform for which you are configuring metrics.

4. Click the + icon . An panel called "New Rule" and the next incremented number opens.

5. Provide the name and settings for this rule:

- The **Name** is the identifier you want to display in the UI.
- An **Exclude Rule** is used for excluding existing rules, so leave the default **No** option.
- **Enabled** means that you want this rule to run, so leave it selected.
- The **Metric Path** is the category as shown in the Metric Browser where the metrics will be displayed. A metric path groups the metrics and is relative to the Metric Browser node.

For example, the following screenshot displays how the JMX Metric Rule "Tomcat\_HttpThreadPools" is defined for the ACME Online demo. The metric path is "Web Container Runtime", the category on Metric Browser where all metrics configured under the "Tomcat\_HttpThreadPools" Metric Rule will be available.

**New Rule 4**

Name	Tomcat_HttpThreadPools
Exclude Rule	<input type="radio"/> Yes <input checked="" type="radio"/> No
Enabled	<input checked="" type="checkbox"/>
Metric Path	Web Container Runtime

This is the path in the metric browser where this metric will be created relative to the JMX metric browser node.

Metric Tree    Settings

- Overall Application Performance
- Business Transaction Performance
- ▼ Application Infrastructure Performance
  - E-Commerce
  - Agent
  - Hardware Resources
  - Individual Nodes
  - ▼ JMX
    - Sessions
    - ▼ Web Container Runtime
    - http-8000
    - JVM
    - Inventory
    - Order Processing

6. Click + icon. An panel called "New Rule" and the next incremented number opens. Specify details for the MBeans that you want to monitor.

- The **Domain name** is the Java domain. This property must be the exact name; no wildcard characters are supported.
- The **Object Name Match Pattern** is the full object name pattern. The property may contain wildcard characters, such as the asterisk for matching all the name/value pairs.
- The **Advanced MBean Matching Criteria** is optional for more complex matching. Use one of the following:
  - any-substring
  - final-substring
  - equals
  - initial-substring

For example, the following screenshot displays the MBean matching criteria for the "Tomcat\_HTTPThreadPools" rule.

The screenshot shows the 'MBeans' configuration screen. At the top, there's a section titled 'MBean Matching Criteria' with fields for 'Domain' (set to 'Catalina') and 'Object Name Match Pattern' (set to 'Catalina:type=ThreadPool,\*'). Below this, a section titled 'Advanced MBean Matching' is expanded, showing a condition builder. The condition is set to 'All' (selected from a dropdown) and contains a single condition: 'name any-substring http'. There is also an 'Add Condition' button with a plus sign.

For all MBeans that match the preceding criteria, you can define one or more metrics for the attributes of those MBeans.

7. In the Attributes panel click **Add Attribute** to specify the MBean attributes.

- Provide the name of the attribute and the metric name.  
The metric name will be used to represent the metric in the AppDynamics metric browser.
- Specify any of the following "Advanced Properties" for the attribute:
  - **Metric Time Rollup** determines how the metric will be aggregated over a period of time. You can choose to either average or sum the data points, or use the latest data point in the time interval.
  - **Metric Cluster Rollup** defines how the metric will be aggregated for a tier, using the performance data for all the nodes in that tier. You can either average or sum the data.
  - **Metric Aggregator Rollup** defines how the Agent rolls up multiple individual measurements (observations) into the observation that it reports once a one minute. For performance reasons, Agents report data to the Controller at one minute intervals. Some metrics, such as Average Response Time, are measured (observed) many times in a minute. The Metric Aggregator Rollup setting determines how the Agent aggregates these metrics. You can average or sum observations on the data points or use the current observation. Alternatively you can use the delta between the current and previous observation (*new in 3.4*).

The following screenshot shows how the MBean attributes configured for the Tomcat\_HttpThreadPools rule will be displayed in the Metric Browser.

The screenshot shows the 'Attributes' section of the MBean Browser. It displays three rows of configuration for defining metrics from MBean attributes:

- Row 1:** MBean Attribute: maxThreads; Metric Name: Maximum Threads.
- Row 2:** MBean Attribute: currentThreadsBusy; Metric Name: Busy Threads.
- Row 3:** MBean Attribute: currentThreadCount; Metric Name: Current Threads In Pool.

Each row has an 'Advanced' button and a red minus sign icon. Below these rows is a green plus sign icon labeled 'Add Attribute'. A large blue arrow points from the right side of the screen towards the 'Metrics' section.

**Metric Tree (Right Side):**

- Metric Tr... (selected)
- Overall Application Performance
- Business Transaction Performance
- Application Infrastructure Performance
  - E-Commerce
    - Agent
    - Hardware Resources
    - Individual Nodes
  - JMX
    - Sessions
    - Web Container Runtime
      - http-8000
        - Busy Threads
        - Current Threads In Pool
        - Error Count
        - Maximum Threads
        - Request Count
        - Total Bytes Received
        - Total Bytes Sent

## Using the MBean Browser

Sometimes you know exactly which particular MBean attribute you want to monitor. You can select the attribute in the MBean Browser and create a JMX Metric Rule for it.

### To create a metric from an MBean attribute in the MBean Browser

1. Navigate to the attribute and select it in the MBean Browser.

The screenshot shows the AppDynamics MBean Browser interface. The top navigation bar includes tabs for Overview, Hardware, Memory, JVM, JMX (which is selected), Problems, and Request Snapshots. Below the navigation is a toolbar with Hide Tree, Refresh Domains, and search icons. The main area displays the MBean tree on the left and its attributes on the right.

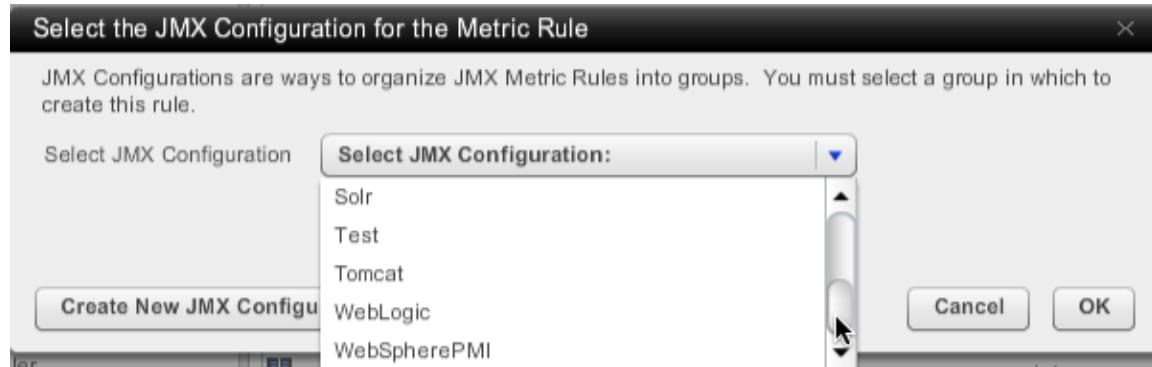
**MBean**  
Object Name: Catalina:type=Connector,port=8000

**Attributes**

	Name	Type
acceptCount	int	
bufferSize	int	
compression	java.lang.S	
connectionLinger	int	
connectionTimeout	int	
connectionUploadTimeout	int	

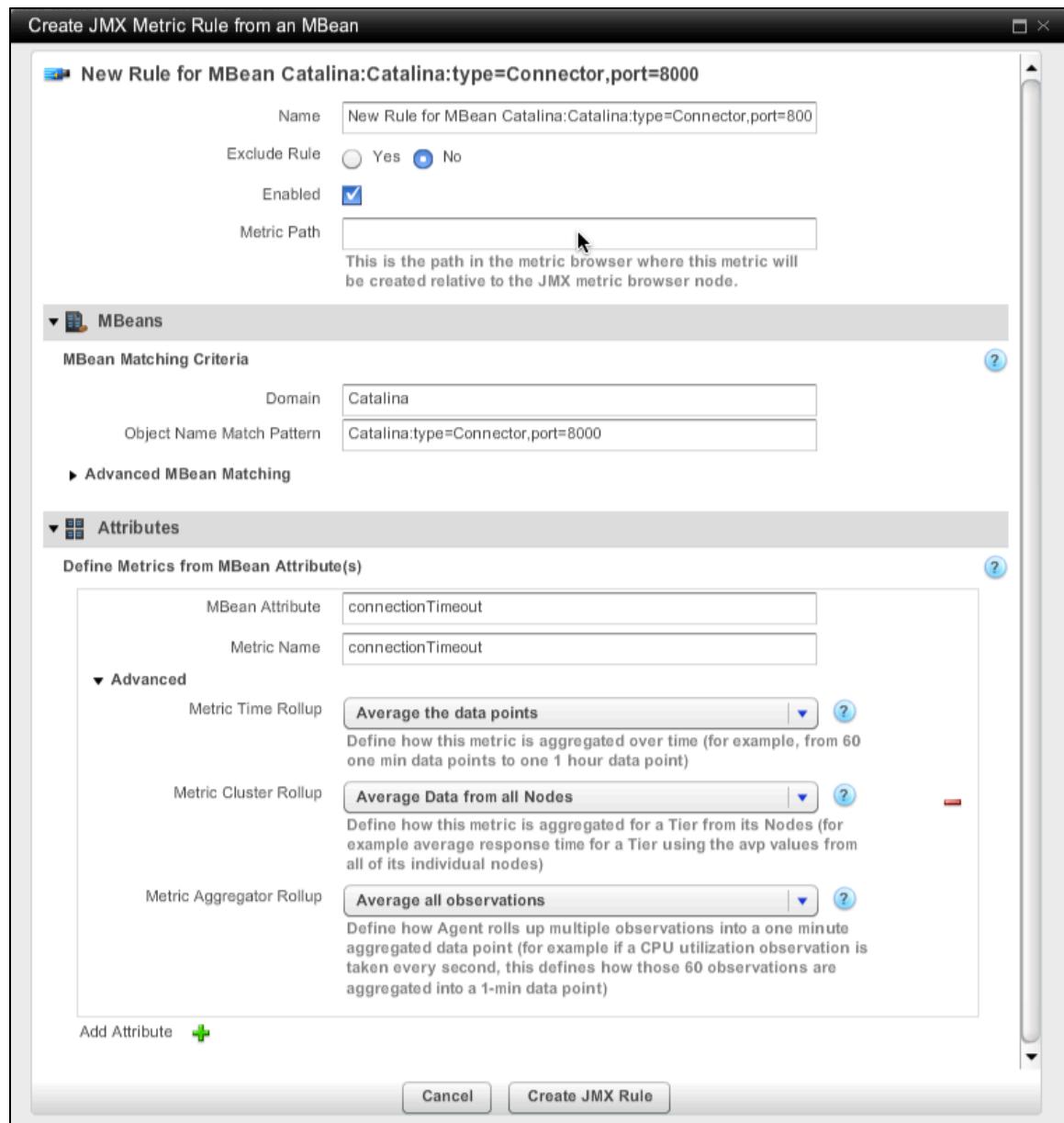
2. Click **Create Metric**.

3. In the **Select JMX Configuration for the Metric Rule** window, select the group in which to create the rule from the **Select JMX Configuration** pulldown. The categories that already exist on your system are listed.



Alternatively, you can create a new group with an name of your choosing. Click **Create New JMX Configuration** to make a new category. This is useful if you want to separate out the custom metrics from the out-of-the-box metrics.

The Create JMX Metric Rule from an MBean window opens.



4. Supply the **Metric Path**, the category as shown in the Metric Browser where the metrics will be displayed. For more discussion of the Metric Path see Step 5 of the previous section.
5. Review the **MBean Matching Criteria** and modify it as needed. Since this is the MBean you selected, you probably do not need to change it.
6. Review the **MBean Attribute** and **Metric Name**. This is the MBean attribute you originally selected. By default the name is the same as the attribute. You can change it if you want to be more specific about its use.
7. Review the **Advanced** panel rollup criteria and update as needed. For more description of these options see Step 7 of the previous section.
8. Click **Create JMX Rule**. The new metric displays in the JMX Metric Browser.

## Learn More

- Monitor JVMs
- Monitor JMX MBeans

- Exclude JMX Metrics

## Exclude JMX Metrics

- Tuning What Metrics are Gathered
  - To exclude a metric
- Learn More

This topic describes how to exclude MBean attributes from being monitored as JMX metrics.

For background information about JMX metrics see [Monitor JVMs](#) and [Monitor JMX MBeans](#).

### Tuning What Metrics are Gathered

AppDynamics provides a default configuration for certain JMX metrics. However, in situations where an environment has many resources, there may be too many metrics gathered. AppDynamics lets you exclude resources and particular operations on resources.

#### To exclude a metric

For example, suppose you want to exclude monitoring for HTTP Thread Pools. Create a new JMX Metrics Rule with the following criteria:

1. Set the Exclude Rule option to **Yes**.
2. Provide the **Object Name Match Pattern**:

```
Catalina:type=ThreadPool,*
```

2. Provide the Advanced MBean Pattern Matching value:

```
http
```

This configuration causes AppDynamics to stop monitoring metrics for HTTP Thread Pools.

Later, if you want to see all HTTP Thread Pool metrics, clear the Enabled checkbox to disable the rule.

#### Learn More

- [Monitor JVMs](#)
- [Monitor JMX MBeans](#)
- [Configure JMX Metrics from MBeans](#)

## Exclude MBean Attributes

- Excluding MBean Attributes from the MBean Browser
  - To exclude an MBean attribute
- Learn More

### Excluding MBean Attributes from the MBean Browser

Some MBean attributes contain sensitive information that you do not want the Java Agent to report. You can configure the Java Agent to exclude these attributes using the <exclude object-name> setting in the app-agent-config.xml file.

#### To exclude an MBean attribute

1. Open the AppServerAgent/conf/app-agent-config.xml file.
2. Locate the JMXService section:

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

3. In the JMXService <configuration> section add the <jmx-metric-browser-excludes> section and the <exclude object-name> property as per the instructions in the comment.

```
<configuration>
    <!--
        Use the below configuration sample to create rules to exclude MBean attributes
        from MBean Browser.
        <exclude object-name=<MBean name pattern> attributes=< * |comma separated list
        of attribute names> >
            The example below will exclude all attributes of MBeans that match
            "Catalina:*".
            <jmx-metric-browser-excludes>
                <exclude object-name="Catalina:/" attributes="*"/>
            </jmx-metric-browser-excludes>
        -->
</configuration>
```

4. Save the file.

The new configuration takes effect immediately if the agent-overwrite property is set to true in the app-agent-config.xml. If agent-overwrite is false, which is the default, then the new configuration will be ignored and you have to restart the agent.

## Learn More

- [App Agent for Java Directory Structure](#)

## Configure JMX Without Transaction Monitoring

- Collect Metrics without Transaction Monitoring
  - To turn off transaction detection
- [Learn More](#)

### Collect Metrics without Transaction Monitoring

In some circumstances, such as for monitoring caches and message buses, you want to collect JMX metrics without the overhead of transaction monitoring.

You can do this by turning off transaction detection at the entry point.

#### To turn off transaction detection

1. In the left navigation panel, click **Configure -> Instrumentation**.
2. In the Select Application or Tier panel, select the application.
3. In the right panel, click **Java Transaction Detection**.
4. Expand the Entry Points list if it is not already expanded.
5. Clear all the relevant **Enabled** checkboxes in the Transaction Monitoring column.

Transaction monitoring on all selected entry points in the application is disabled. Exit point detection remains enabled.

## Learn More

- [Configure Business Transaction Detection](#)

## Resolve JMX Configuration Issues

- [Unable to browse MBeans on WebSphere Application Server \(WAS\)](#).

- Unable to get metrics from the Java App Server Agent on GlassFish
  - Unable to get JMX metrics for database connections on GlassFish
- Learn More

This topic describes how to resolve issues that may prevent AppDynamics from properly reporting JMX MBean metrics.

### Unable to browse MBeans on WebSphere Application Server (WAS).

In certain situations, you may encounter the following exception in the agent.log file for a Java App Server Agent deployed on the WebSphere Application Server (WAS).

```
[AD Thread-Transient Event Channel Poller0] 17 Aug 2011 08:14:08,031
ERROR JMXTransientOperationsHandler - Error trying to lookup clz -
java.lang.ClassNotFoundException: com.ibm.ws.security.core.SecurityContext
```

To resolve this issue:

1. From the WAS administration console, navigate to the JVM settings for the server of interest: **Application servers -> <server> -> Process Definition -> Java Virtual Machine**.
2. Remove the following setting from the generic JVM settings:

```
-Djavax.management.builder.initial = -Dcom.sun.management.jmxremote
```

### Unable to get metrics from the Java App Server Agent on GlassFish

Under some situations JMX metrics from GlassFish are not reported. Also some metrics may not be enabled by default. Try these solutions:

- 1) Confirm that JMX monitoring is enabled in the GlassFish server. Refer to the following screenshot:

- 2) Copy the text below into an mbean-servers.xml file in the following directory:

```
<App_Agent_Dir>/conf/jmx/
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--<!DOCTYPE servers SYSTEM "mbean-servers.dtd"> -->

<servers xmlns="http://www.appdynamics.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.appdynamics.com
  mbean-servers.xsd">
  <!--
    <server mbean-server-name="WebSphere"
    mbean-name-pattern="WebSphere:*,type=Server,j2eeType=J2EEServer"
    version-attribute="platformVersion" version-startsWith="7"
    config-file="servers/websphere-7-jmx-config.xml" />

    <server mbean-server-name="WebSphere"
    mbean-name-pattern="WebSphere:*,type=Server,j2eeType=J2EEServer"
    version-attribute="platformVersion" version-startsWith="6"
    config-file="servers/websphere-7-jmx-config.xml" />
  -->
  <server mbean-server-name="WebSphere"
    mbean-name-pattern="WebSphere:*,type=Server"
    config-file="servers/websphere-7-jmx-config.xml" />
    <server mbean-server-name="JBoss_4"
    mbean-name-pattern="jboss.management.local:j2eeType=J2EEServer,name=Local"
    version-attribute="serverVersion" version-startsWith="4"
    config-file="servers/jboss-4-jmx-config.xml" />
    <server mbean-server-name="JBoss_5"
    mbean-name-pattern="jboss.management.local:j2eeType=J2EEServer,name=Local"
    version-attribute="serverVersion" version-startsWith="5"
    config-file="servers/jboss-5-jmx-config.xml" />
    <server mbean-server-name="JBoss_6"
    mbean-name-pattern="jboss.management.local:j2eeType=J2EEServer,name=Local"
    version-attribute="serverVersion" version-startsWith="6"
    config-file="servers/jboss-5-jmx-config.xml" />
    <server mbean-server-name="Tomcat_5.5"
    mbean-name-pattern="Catalina:type=Server" version-attribute="serverInfo"
    version-startsWith="Apache Tomcat/5.5"
    config-file="servers/tomcat-5-jmx-config.xml" />
    <server mbean-server-name="Tomcat_6.0"
    mbean-name-pattern="Catalina:type=Server" version-attribute="serverInfo"
    version-startsWith="Apache Tomcat/6.0"
    config-file="servers/tomcat-6-jmx-config.xml" />
    <server mbean-server-name="Tomcat_7"
    mbean-name-pattern="Catalina:type=Server" version-attribute="serverInfo"
    version-startsWith="Apache Tomcat/7"
    config-file="servers/tomcat-7-jmx-config.xml" />
    <server mbean-server-name="Sun GlassFish_2.1"
    mbean-name-pattern="com.sun.appserv:j2eeType=J2EEServer,name=server,category=runt
    config-file="servers/glassfish-v2-jmx-config.xml" />
    <server mbean-server-name="WebLogic_10"
    mbean-server-lookup-string="java:comp/jmx/runtime"
```

```

mbean-name-pattern="com.bea:*,Type=ServerRuntime"
version-attribute="WeblogicVersion" version-startsWith="WebLogic Server 10"
config-file="servers/weblogic-10-jmx-config.xml" />
    <server mbean-server-name="WebLogic_9"
mbean-server-lookup-string="java:comp/jmx/runtime"
mbean-name-pattern="com.bea:*,Type=ServerRuntime"
version-attribute="WeblogicVersion" version-startsWith="WebLogic Server 9"
config-file="servers/weblogic-9-jmx-config.xml" />
    <server mbean-server-name="ActiveMQ_5.3.2"
mbean-name-pattern="org.apache.activemq:*"
config-file="servers/activemq-5.3.2-jmx-config.xml" />
    <server mbean-server-name="Apache Solr 1.4.1" mbean-name-pattern="solr:*"
config-file="servers\solr-1.4.1-jmx-config.xml" />
    <server mbean-server-name="Apache Cassandra 0.7.0"
mbean-name-pattern="org.apache.cassandra.net:*"
config-file="servers\cassandra-0.7.0-jmx-config.xml" />
    <server mbean-server-name="Apache Cassandra 0.7.0"
mbean-name-pattern="org.apache.cassandra.db:*"
config-file="servers\cassandra-0.7.0-jmx-config.xml" />
    <server mbean-server-name="Apache Cassandra 0.7.0"
mbean-name-pattern="org.apache.cassandra.request:*"
config-file="servers\cassandra-0.7.0-jmx-config.xml" />
    <server mbean-server-name="Apache Cassandra 0.7.0"
mbean-name-pattern="org.apache.cassandra.internal:*"
config-file="servers\cassandra-0.7.0-jmx-config.xml" />
    <!-- If you are using Platform MBean server to report activemq metrics then
you may uncomment the following line.
-->
<!--
<server mbean-server-name="Platform"
mbean-name-pattern="org.apache.activemq:*"
config-file="servers\activemq-5.3.2-jmx-config.xml" />
-->

<!-- If your app publishes custom jmx metrics to platform jmx server then
you may modify the platform-jmx-config.xml
and update the mbean-name-pattern in the following line to start recording
your metrics by appdynamics agent
-->

<!--
<server mbean-server-name="Platform" mbean-name-pattern="com.foo.myjmx:*"
config-file="servers\platform-jmx-config.xml" />
-->

```

```
</servers>
```

You should see a new JMX node in the metrics tree.

#### ***Unable to get JMX metrics for database connections on GlassFish***

JDBC connection pool metrics are not configured out-of-the-box for GlassFish. To configure them, uncomment the JDBC connection pool section and provide the relevant information in the following file:

```
<app_agent_install>/conf/jmx/servers/glassfish-v2-jmx-config.xml
```

Uncomment the following section and follow the instructions provided in the file.

```

<!-- The following config can be uncommented to monitor glassfish JDBC
connection pool. Please set the name of the connection
pool (not the datasource name) and enable monitoring for the JDBC Pools on
glassfish admin console. -->
<!--
<metric
mbean-name-pattern="com.sun.appserv:type=jdbc-connection-pool,category=monitor,nat
the name of pool,<*>
category="JDBC Connection Pools">
<attribute-counter-mappings>
<attribute-counter-mapping>
<attribute-name>numconnused-current</attribute-name>
<counter-name>Connections In Use</counter-name>
<counter-type>average</counter-type>
<time-rollup-type>average</time-rollup-type>
<cluster-rollup-type>individual</cluster-rollup-type>
</attribute-counter-mapping>
<attribute-counter-mapping>
<attribute-name>numconnused-highwatermark</attribute-name>
<counter-name>Max Connections Used</counter-name>
<counter-type>observation</counter-type>
<time-rollup-type>average</time-rollup-type>
<cluster-rollup-type>individual</cluster-rollup-type>
</attribute-counter-mapping>
<attribute-counter-mapping>
<attribute-name>numpotentialconnleak-count</attribute-name>
<counter-name>Potential Leaks</counter-name>
<counter-type>observation</counter-type>
<time-rollup-type>average</time-rollup-type>
<cluster-rollup-type>individual</cluster-rollup-type>
</attribute-counter-mapping>
<attribute-counter-mapping>
<attribute-name>averageconnwaittime-count</attribute-name>
<counter-name>Avg Wait Time Millis</counter-name>
<counter-type>observation</counter-type>
<time-rollup-type>average</time-rollup-type>
<cluster-rollup-type>individual</cluster-rollup-type>
</attribute-counter-mapping>
<attribute-counter-mapping>
<attribute-name>waitqueuelength-count</attribute-name>
<counter-name>Current Wait Queue Length</counter-name>
<counter-type>observation</counter-type>
<time-rollup-type>average</time-rollup-type>
<cluster-rollup-type>individual</cluster-rollup-type>
</attribute-counter-mapping>
</attribute-counter-mappings>
</metric>

```

## Learn More

- IBM WebSphere Startup Settings
- Glassfish Startup Settings

# Import or Export JMX Metric Configurations

- To import JMX metrics configuration
- To export JMX metrics configuration
- To use a pre-3.3 configuration file for JMX metrics

This topic describes how to import or export your existing JMX configurations.

## To import JMX metrics configuration

1. Click **Configure -> Instrumentation**.
2. Click the **JMX** tab.
3. Click the **Import JMX Configuration** icon.



4. In the JMX Configuration Import screen, click **Select JMX Config. File** and select the XML configuration file for your JMX metrics.



5. Click **Import**.

## To export JMX metrics configuration

1. Click **Configure -> Instrumentation**.
2. Click the **JMX** tab.
3. Click the **Export JMX Configuration** icon.

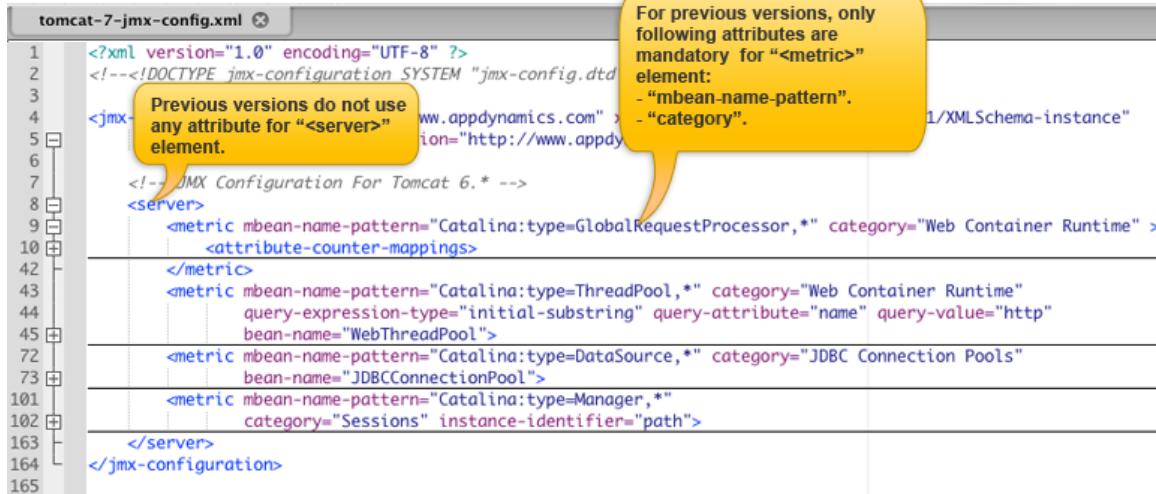


The configuration is downloaded as an XML file.

## To use a pre-3.3 configuration file for JMX metrics

In AppDynamics Pro Version 3.3 the structure of the XML-based configuration file for JMX metrics changed. As a result, if you use a pre-3.3 version, you must provide additional attributes in the configuration file.

The following screenshot shows a sample XML configuration file for the JMX metrics for pre-3.3 versions of AppDynamics:



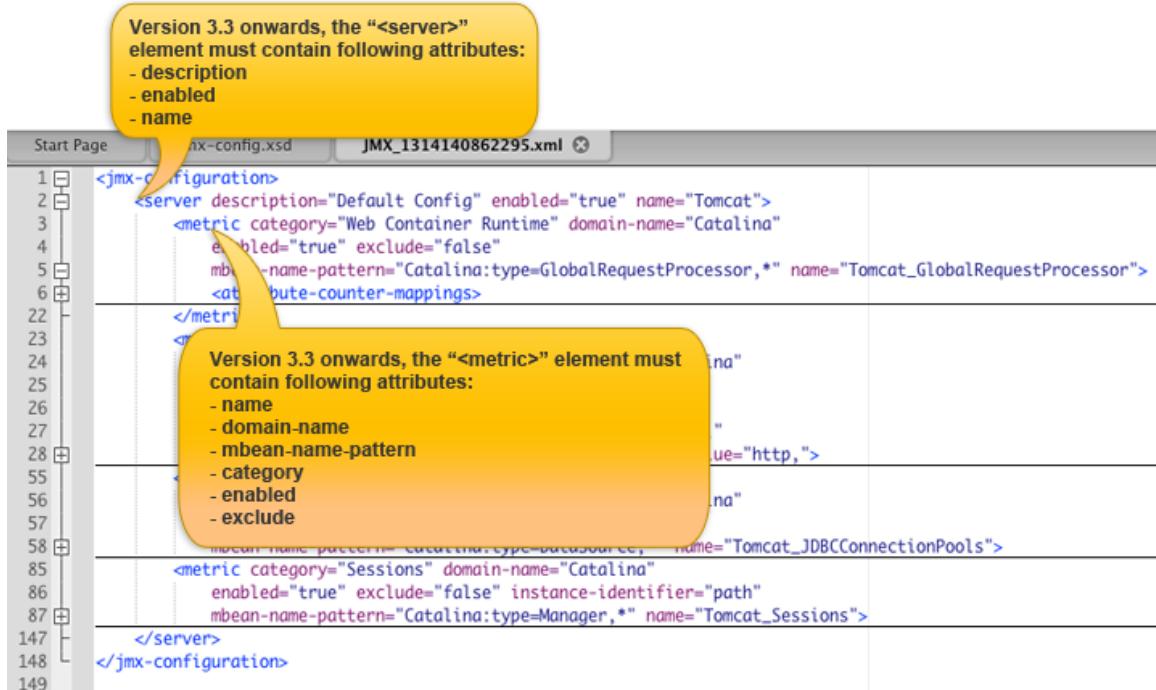
```

tomcat-7-jmx-config.xml

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE jmx-configuration SYSTEM "jmx-config.dtd"
3
4  <jmx-configuration instance-identifier="http://www.appdynamics.com" version="1.0" >
5      Previous versions do not use any attribute for <server> element.
6
7      <!-- JMX Configuration For Tomcat 6.* -->
8      <server>
9          <metric mbean-name-pattern="Catalina:type=GlobalRequestProcessor,*" category="Web Container Runtime" >
10         <attribute-counter-mappings>
11
12     </metric>
13     <metric mbean-name-pattern="Catalina:type=ThreadPool,*" category="Web Container Runtime" >
14         query-expression-type="initial-substring" query-attribute="name" query-value="http"
15         bean-name="WebThreadPool">
16
17     <metric mbean-name-pattern="Catalina:type=DataSource,*" category="JDBC Connection Pools" >
18         bean-name="JDBCConnectionPool">
19
20     <metric mbean-name-pattern="Catalina:type=Manager,*" category="Sessions" instance-identifier="path">
21
22 </server>
23 </jmx-configuration>

```

Beginning with AppDynamics version 3.3, the structure for this XML file is the following:



```

Start Page   jmx-config.xsd   JMX_1314140862295.xml
1  <jmx-configuration>
2      <server description="Default Config" enabled="true" name="Tomcat">
3          <metric category="Web Container Runtime" domain-name="Catalina" >
4              enabled="true" exclude="false"
5                  mbean-name-pattern="Catalina:type=GlobalRequestProcessor,*" name="Tomcat_GlobalRequestProcessor">
6                  <attribute-counter-mappings>
7
8          </metric>
9
10     <metric category="Web Container Runtime" domain-name="Catalina" >
11         enabled="true" exclude="false"
12             mbean-name-pattern="Catalina:type=GlobalRequestProcessor,*" name="Tomcat_GlobalRequestProcessor">
13             <attribute-counter-mappings>
14
15     <metric category="Web Container Runtime" domain-name="Catalina" >
16         enabled="true" exclude="false"
17             mbean-name-pattern="Catalina:type=ThreadPool,*" name="Tomcat_WebThreadPool">
18             <attribute-counter-mappings>
19
20     <metric category="Web Container Runtime" domain-name="Catalina" >
21         enabled="true" exclude="false"
22             mbean-name-pattern="Catalina:type=DataSource,*" name="Tomcat_JDBCConnectionPools">
23             <attribute-counter-mappings>
24
25     <metric category="Sessions" domain-name="Catalina" >
26         enabled="true" exclude="false" instance-identifier="path"
27             mbean-name-pattern="Catalina:type=Manager,*" name="Tomcat_Sessions">
28             <attribute-counter-mappings>
29
30 </server>
31 </jmx-configuration>

```

To be able to import your existing configurations for JMX metrics, add the following attributes for `<server>` element and **each** of the `<metric>` elements in your XML file.

The attributes for each element are listed below:

#### Attributes for the `<server>` element

Attribute Name	Attribute Type	Allowed Values	Mandatory/Optional
description	String		Mandatory
enabled	Boolean	true/false	Mandatory
name	String		Mandatory

#### Attributes for each <metric> element

For each <metric> element, add the mandatory attributes from the following list to your existing configuration file:

Attribute Name	Attribute Type	Allowed Values	Mandatory/Optional
name	String		Mandatory
domain-name	String	true/false	Mandatory
mbean-name-pattern	String		Mandatory
category	String		Mandatory <i>All those &lt;metric&gt; elements that have same value for this attribute will be grouped together on the metric browser.</i>
enabled	Boolean	true/false	Mandatory
exclude	Boolean	true/false	Mandatory
bean-name	String		Optional
query-attribute	String		Optional
query-expression-type		Use any one from the following values: <ul style="list-style-type: none"><li>• any-substring</li><li>• final-substring</li><li>• equals</li><li>• initial-substring</li></ul>	Optional
query-value	String		Optional
instance-identifier	String		Optional
instance-name	String		Optional

## Configure Custom Memory Structures (Java)

- Custom Memory Structures and Memory Leaks
  - Using Automatic Leak Detection vs Monitoring Custom Memory Structures
  - Supported JVMs
    - To identify custom memory structures
    - To Add a Custom Memory Structure
- Identifying Potential Memory Leaks
  - Diagnosing memory leaks
  - Isolating a leaking collection
  - Access Tracking
- Learn More

This topic describes how to configure custom memory structures and monitor the potential leaks in production environments.

## Custom Memory Structures and Memory Leaks

Typically custom memory structures are used as caching solutions. In a distributed environment, caching can easily become a source of memory leaks. AppDynamics helps you to manage and track memory statistics for these memory structures.

AppDynamics provide visibility into:

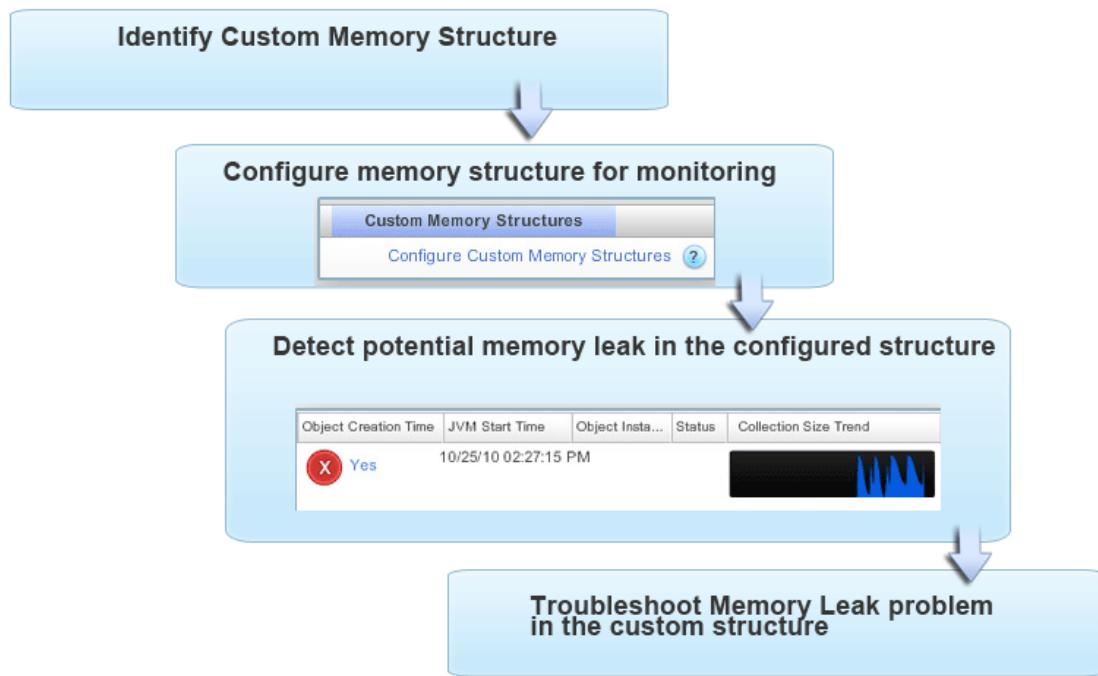
- Cache access for slow, very slow, and stalled business transactions
- Usage statistics, rolled up to the Business Transaction level
- Keys being accessed

- Deep size of internal cache structures

## Using Automatic Leak Detection vs Monitoring Custom Memory Structures

The automatic leak detection feature captures memory usage data for all Collections objects in a JVM. However, custom memory structures might or might not contain Collections objects. For example, you can have a custom cache or a third party cache like Ehcache about which you can collect memory usage statistics.

The following illustration provides the work flow for configuring, monitoring, and troubleshooting custom memory structures. You have to configure custom memory structures manually.



## Supported JVMs

The supported JVMs for the AppDynamics custom memory monitoring feature are:

- Sun JVM 1.51
- BEA JRockit JVM 1.5 (requires a JVM restart)
- IBM JVM 1.5 (content inspection functionality only)
- Sun JVM 1.6
- BEA JRockit JVM 1.6 (content inspection functionality only)
- IBM JVM 1.6 (content inspection functionality only)

### To identify custom memory structures

Enable On-demand Access Tracking in automatic leak detection to capture information on what classes are accessing which Collections objects. Use this information to identify custom memory structures.

AppDynamics captures the top 1000 classes, by instance count.

### To Add a Custom Memory Structure

1. From the left navigation pane select **Configure -> Instrumentation**.
2. Click the **Memory Monitoring** tab.
3. In the Tier panel select the tier for which you want to configure a custom memory structure.

4. In the Custom memory Structures section click **Add** to add a new memory structure. The Add Memory Structure window opens.

5. In the Create Memory Structure window

- Specify the configuration name.
- Check **Enabled**.

6. Specify the discovery method.

The discovery method provides three options to monitor the Custom Memory Structure. The discovery method determines how the agent gets a reference to the Custom Memory Structure. AppDynamics needs this reference to monitor the size of the structure. Select any of the three options for the discovery method:

- Discover using Static Field.
- Discover using Constructor.
- Discover using Method.

In many cases, especially with caches, the object for which a reference is needed is created early in the life cycle of the application.

Example for using static field	Example for using Constructor	Example for using method
<pre>public class CacheManager { private static Map&lt;String&gt; userCache&lt;String&gt; User; }</pre>	<pre>public class CustomerCache { public CustomerCache(); }</pre>	<pre>public Class CacheManager{ public List&lt;Order&gt; getOrderCache(); { } }</pre>

Notes: Monitors deep size of this Map.

Notes: Monitors deep size of CustomerCache object(s).

Notes: Monitors deep size of this list.

Restart the JVM after the discovery methods are configured to get the references for the object.

5. (Optional) Define accessors.

Click **Define Accessors** to define the methods used to access the custom memory structure. This information is used to capture the code paths accessing the custom memory structure.

6. (Optional) Define the naming convention.

Click **Define Naming Convention**. These configurations differentiate between custom memory structures.

There are situations where more than one custom Caches are used, but only few of them need monitoring. In such a case, use the **Getter Chain** option to distinguish amongst such caches. For all other cases, use either value of the field on the object or a specific string as the object name.

6. Click **Save** to save the configuration.

## Identifying Potential Memory Leaks

Start monitoring memory usage patterns for custom memory structures. An object is automatically marked as a potentially leaking object when it shows a positive and steep growth slope. The Memory Leak Dashboard provides the following information:

- **Collection Size:** It provides the number of elements in a Collection.
- **Potentially Leaking:** Potentially leaking Collections are marked as red. It is recommended to start diagnostic sessions on potentially leaking objects.
- **Status:** Indicates if a diagnostic session has been started on an object.
- **Collection Size Trend:** A positive and steep growth slope indicates potential memory leak.



Identify long-lived Collections by comparing the JVM start time and Object Creation Time. After the potentially leaking Collections are identified, start the diagnostic session.

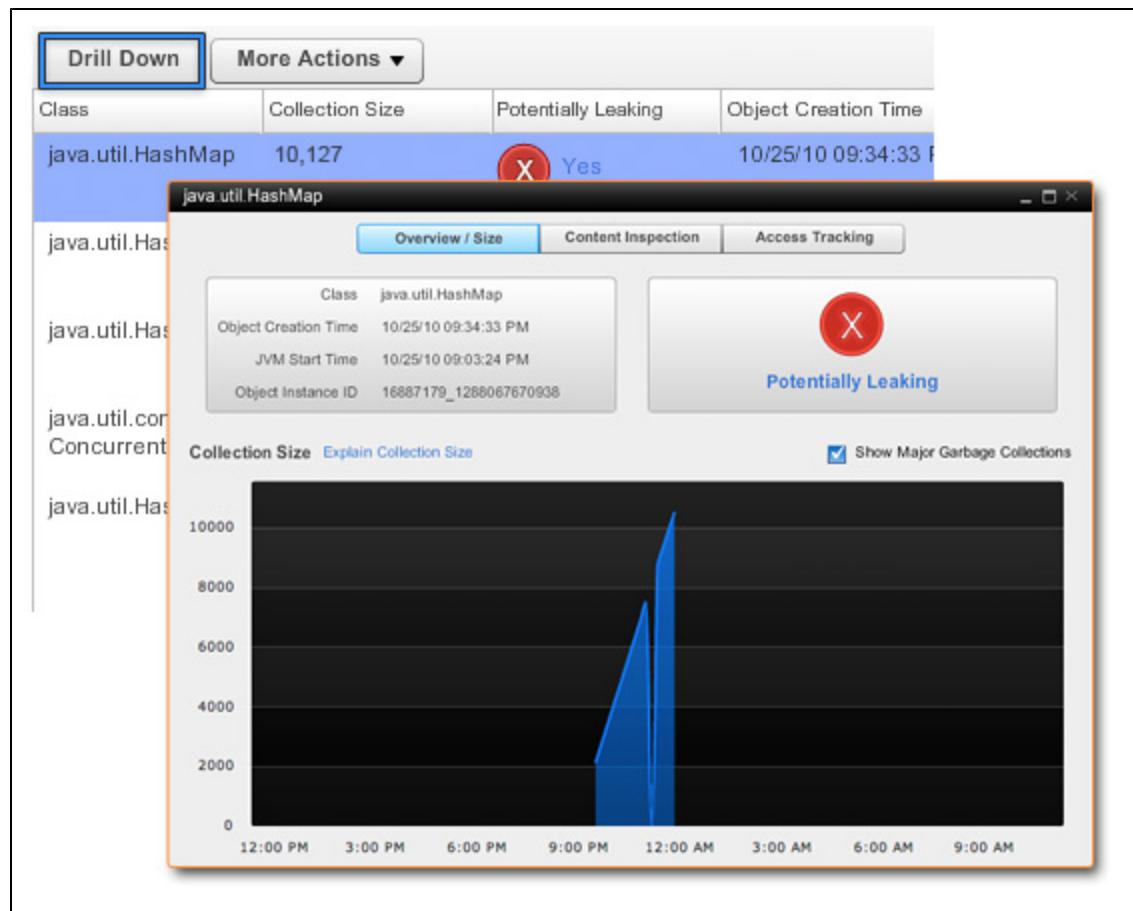
## Diagnosing memory leaks

Select the class name to monitor and click **Drill Down** or right-click on the class name and select **Drill Down**.

### Isolating a leaking collection

Use Content Inspection to identify to which part of the application the Collection belongs. It allows monitoring histograms of all the elements in a particular memory structure. Start a diagnostic session on the object and then follow these steps:

1. Select **Content Inspection**.
2. Click **Start Content Summary Capture Session**.
3. Enter the session duration. Allow at least 1-2 minutes for the data to generate.
4. Click **Refresh** to retrieve the session data.
5. Click on the particular snapshot to view the details about an individual session.



## Access Tracking

Use Access Tracking to view the actual code paths and business transactions accessing the memory structure. Start a diagnostic session on the object and follow these steps:

1. Select the "Access Tracking".
2. Select "Start Access Tracking Session".
3. Enter the session duration. Allow at least 1-2 minutes for data generation.
4. Click the refresh button to retrieve the session data.
5. Click on the particular snapshot, to view the details about an individual session.

## Learn More

- Troubleshoot Java Memory Leaks

## Configure Object Instance Tracking (Java)

- Prerequisites for Object Instance Tracking
  - To enable object instance tracking on a node
- Tracking Specific Classes
  - To track instances of custom classes
- Learn More

This topic helps you understand how to configure object instance tracking. For more information about why you may need to configure

this, see [Troubleshoot Java Memory Thrash](#).

## Prerequisites for Object Instance Tracking

- Object Instance Tracking can be used only for Sun JVM v1.6.x and later.
- If you are running with the JDK then tools.jar will already be in the CLASSPATH, but if you are running with the JRE, you must add tools.jar to the CLASSPATH and restart the JVM for this feature to start working.

Adding tools.jar file:

For AppDynamics to read the tools.jar, add this file to jre/lib/ext directory and to the classpath as shown below (along with other -jar options):

```
java -classpath <complete-path-to-tools.jar>%classpath% -jar myApp.jar  
OR  
java -classpath <complete-path-to-tools.jar>:$classpath -jar myApp.jar
```

### To enable object instance tracking on a node

1. In the left navigation pane, click **Servers** -> **App Servers** -> **<tier>** -> **<node>**. The Node Dashboard opens.
2. Click the Memory tab.
3. Click the Object Instance Tracking subtab.
4. Click the ON button.

The screenshot shows the AppDynamics Node Dashboard for the 'Node\_8000' node. The 'Memory' tab is selected. Under the 'Object Instance Tracking' subtab, the 'On' button is highlighted. A tooltip message states: "AppDynamics automatically tracks the top 20 application classes, and the top 20 system (core Java) classes in the heap by number of instances. In addition to these top classes, you can configure any specific classes that you want to track by clicking on 'Configure Custom Classes to Track'. You can drill down into any of these classes and track what code paths are allocating instances of them."

## Tracking Specific Classes

Check against the required set of classes to enable instance tracking for each set. For improved performance, only the top 20 classes are tracked.

Use Configure Custom Classes to track instances of specific classes.

Classes configured here will only be tracked if their instance count is among the top 1000 instance counts in the JVM.

### To track instances of custom classes

1. In the left navigation pane, click **Servers** -> **App Servers** -> **<tier>** -> **<node>**. The Node Dashboard opens.
2. Click **Configure Custom Classes To Track** on the rightmost corner of the screen.
3. Click **Add**.
4. Click **Enabled**.

## Learn More

- Troubleshoot Java Memory Thrash

## Configure Memory Monitoring (Java)

- Prerequisites for Object Instance Tracking
  - To enable object instance tracking
- Tracking Specific Classes
  - To Specify Custom Classes to Track
- Learn More

This topic helps you understand how to configure memory monitoring, also known as object instance tracking. For more information about why you may need to configure this, see [Troubleshoot Java Memory Thrash](#).

## Prerequisites for Object Instance Tracking

Object Instance Tracking can be used only for Sun JVM v1.6.x and later.

If you are running with the JDK, tools.jar will already be in the CLASSPATH, but if you are running with the JRE, you must add tools.jar to the CLASSPATH and restart the JVM for this feature to start working.

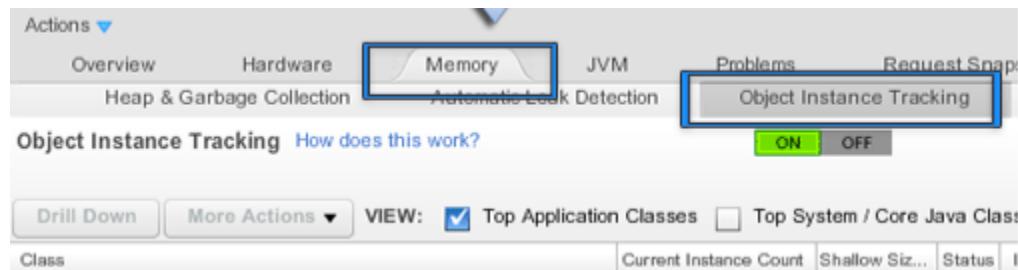
Adding tools.jar file:

For AppDynamics to read the tools.jar, add this file to jre/lib/ext directory and to the classpath as shown below (along with other -jar options):

```
java -classpath <complete-path-to-tools.jar>%classpath% -jar myApp.jar  
OR  
java -classpath <complete-path-to-tools.jar>:$classpath -jar myApp.jar
```

### To enable object instance tracking

1. Navigate to the node dashboard of the node for which you want to enable object instance tracing..
2. Click the **Memory** tab.
3. Select **Object Instance Tracking**.
4. Turn the object instance tracking mode ON.



## Tracking Specific Classes

Check against the required set of classes to enable instance tracking for each set. For improved performance, only the top 20 classes are tracked.

VIEW:  Top Application Classes  Top System / Core Java Classes  Custom Classes  All

Use Configure Custom Classes to track instances of specific classes.

Classes configured here will only be tracked if their instance count is among the top 1000 instance counts in the JVM.

## To Specify Custom Classes to Track

1. Click **Configuration -> Instrumentation -> Memory Monitoring**.
2. Select the tier in which to configure object tracking.
3. In the Object Instance Tracking. Define Custom Classes to Track section, click **Add**.
4. In the Create New Instance Tracker window, check **Enabled**.
5. Enter the fully qualified class name of the class to track.
6. Click **Save**.

You can edit or delete the object tracing configuration after it has been created.

## Learn More

- [Troubleshoot Java Memory Thrash](#)

## Configure Multi-Threaded Transactions (Java)

- [Default Configuration](#)
- [Custom Configuration](#)
  - [Managing Thread Correlation Using a Node Property](#)
- [Enabling and Disabling Asynchronous Monitoring](#)
  - [To Disable Asynchronous Monitoring](#)
  - [To Enable Asynchronous Monitoring](#)
- [Learn More](#)

AppDynamics collects and reports key performance metrics for individual threads in multi-threaded Java applications. See [Trace Multi-Threaded Transactions \(Java\)](#) for details on where these metrics are reported.

## Default Configuration

Classes for multi-threaded correlation are configured in the <excludes> child elements of the <fork-config> element in the <App\_Server\_Agent\_Installation\_Directory>/conf/app-agent-config.xml file.

The default configuration excludes the java, org, weblogic and websphere classes:

```
<fork-config>
    <!-- exclude java and org -->
    <excludes filter-type="STARTSWITH"
filter-value="java/, javax/, com.sun/, sun/, org/" />
    <!-- exclude weblogic and websphere -->
    <excludes filter-type="STARTSWITH"
filter-value="com.bea/, com.weblogic/, weblogic/, com.ibm/" />
    . . .

```

## Custom Configuration

You can edit the app-agent-config.xml file to exclude additional classes from thread correlation. All classes not excluded are by default included.

You can also explicitly include sub-packages and subclasses of excluded packages and classes. Use the <includes> or <include> child element to specify the included packages and classes.

Use the <excludes> and <includes> elements to specify a comma-separated list of classes or packages. Use the <exclude> and <include> elements to specify a single class or package

## Managing Thread Correlation Using a Node Property

You can also configure which classes or packages to include or exclude using a node property. See [thread-correlation-classes](#) and [thread-correlation-classes-exclude](#).

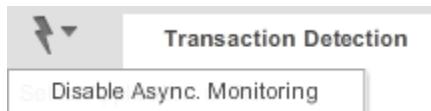
## Enabling and Disabling Asynchronous Monitoring

 You should disable monitoring of multi-threaded transactions on all agents if all of your agents and your controller are not at AppDynamics version 3.6 or higher. You can enable the feature after all of your agents have been upgraded.

You must restart the agent after you enable or disable this feature.

### To Disable Asynchronous Monitoring

1. In the left navigation pane click **Configure->Instrumentation->Transaction Detection**.
2. From the Actions menu in the upper left corner click **Disable Async Monitoring**.



### To Enable Asynchronous Monitoring

1. In the left navigation pane click **Configure->Instrumentation->Transaction Detection**.
2. From the Actions menu in the upper left corner click **Enable Async Monitoring**.



## Learn More

- [Configure Business Transaction Detection](#)
- [App Agent Node Properties](#)

## Configure Background Tasks (Java)

- Pre-Configured Frameworks for Java Background Tasks
- Enabling Automatic Discovery for Background Tasks
  - To enable discovery for a background task using a common framework
- Configuring Background Batch or Shell Files using a Main Method
  - To instrument the main method of a background task
- [Learn More](#)

In a Java environment background tasks are detected using POJO entry points. It is the same basic procedure as defining entry points for business transactions, except that you check the Background Task check box. For instructions see [Configure Background Tasks](#).

## Pre-Configured Frameworks for Java Background Tasks

When enabled, AppDynamics provides discovery for the following Java background-processing task frameworks:

- Quartz
- Cron4J
- JCronTab
- JavaTimer

## Configure Background Tasks

### Enabling Automatic Discovery for Background Tasks

Automatic discovery of background tasks is disabled by default. When you know that there are background tasks in your application environment and you want to monitor them, first enable automatic discovery so that AppDynamics will detect the task.

AppDynamics provides preconfigured support for some common frameworks. If your application is not using one of the default frameworks you can create a custom match rule.

#### To enable discovery for a background task using a common framework

1. In the left navigation pane, click **Configure -> Instrumentation**.
2. On the Transaction Detection tab, select the tier for which you want to enable monitoring.
3. Click **Use Custom Configuration for this Tier**.
4. Scroll down to the Custom Match Rules pane.
5. Do one of the following
  - If you are using a pre-configured framework, select the row of the framework and click the pencil icon, or double-click on the row to open the Business Transaction Match Rule window. By default the values are populated with rule name and the class and method names for the particular framework. Verify that those are the correct names for your environment.  
OR
  - If you are using a custom framework, select the match criteria and enter the Class Name and Method Name.

The Background Task check box should be already checked.

6. Check **Enabled**.

7. Click **Save**.

The custom match rule for the background task will take effect and the background task will display in the Business Transaction List.

Once you enable discovery, every background task is identified based on following attributes:

- Implementation class name
- Parameter to the execution method name

### Configuring Background Batch or Shell Files using a Main Method

Sometimes background tasks are defined in batch or shell files in which the main method triggers the background processing. In this situation, the response time of the batch process is the duration of the execution of the main method.

**⚠️ IMPORTANT:** Instrument the main method only when the duration of the batch process is equal to the duration of the main method. Otherwise choose another method that accurately represents the unit of work for the background process.

#### To instrument the main method of a background task

1. In the left navigation pane, click **Configure -> Instrumentation**.
2. In the **Transaction Detection** section select the tier for which you want to instrument the main method.
3. In the **Custom Rules** section click **Add** (the "+" icon).

4. From the **Entry Point** type drop down list, click **POJO**.

5. Enter a name for the custom rule.

6. Check **Background Task**.

7. Check **Enabled**.

8. Enter "main" as the match value for **Method Name**.

New Business Transaction Match Rule - POJO

Name	OvernightBatchRun
Enabled	<input checked="" type="checkbox"/>
Background Task	<input checked="" type="checkbox"/>

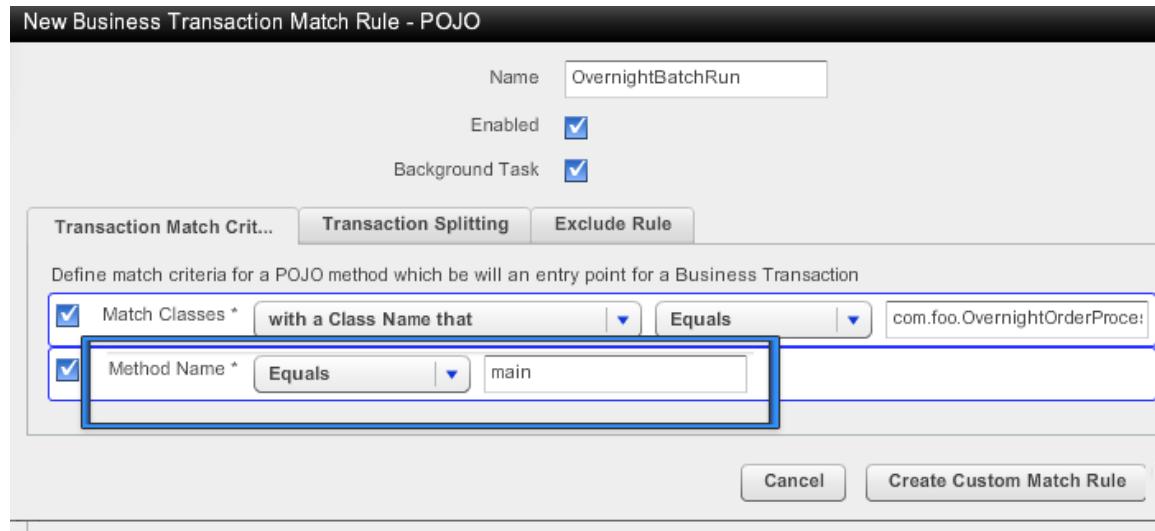
Transaction Match Crit...    Transaction Splitting    Exclude Rule

Define match criteria for a POJO method which will be an entry point for a Business Transaction

Match Classes \* with a Class Name that Equals com.foo.OvernightOrderProce:

Method Name \* Equals main

Cancel    Create Custom Match Rule



9. Save the changes.

10. To ensure that the name of the script file is automatically picked up as a background task, configure your Java Agent for that node. See [Configure App Agent for Java for Batch Processes](#).

## Learn More

- [Configure Background Tasks](#)
- [Configure App Agent for Java for Batch Processes](#)
- [POJO Entry Points](#)

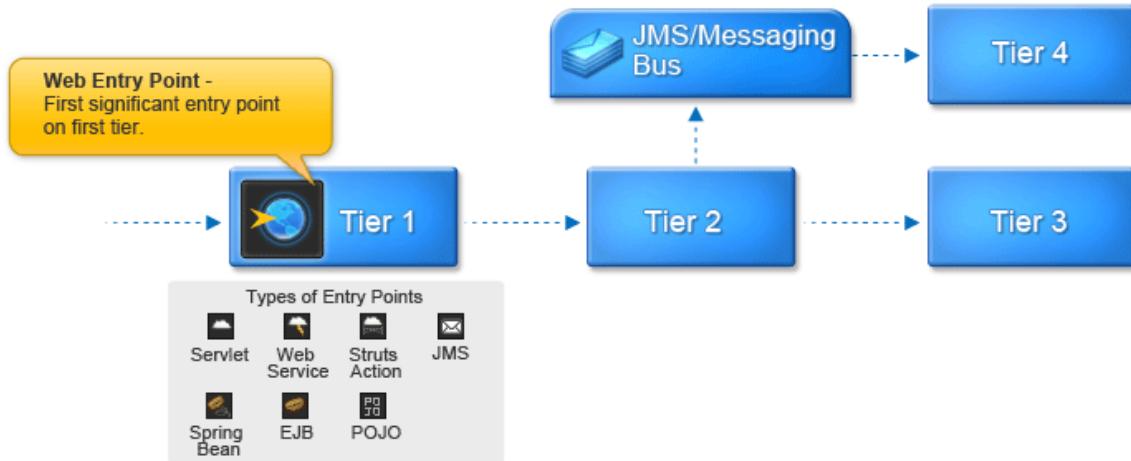
## Web Application Entry Points

- [Tiers and Web Application Entry Points](#)
- [Other Web Application Frameworks Based on Servlets or Servlets Filter](#)
- [Learn More](#)

This section discusses web application entry points for different types of business transactions.

## Tiers and Web Application Entry Points

A tier can have multiple entry points.



For example, for Java frameworks a combination of pure Servlets or JSPs, Struts, Web services, Servlet filters, etc. may all co-exist on the same JVM.

The middle-tier components like EJBs and Spring beans are usually not considered Entry Points because they are normally accessed using either the front-end layers such as Servlets or from classes that invoke background processes.

## Other Web Application Frameworks Based on Servlets or Servlets Filter

AppDynamics provides out-of-the-box support for most of the common web frameworks that are based on Servlets or Servlet Filters. When using any of the frameworks listed below, refer to the [Servlet discovery rules](#) to configure transaction discovery.

- Spring MVC
- Wicket
- Java Server Faces (JSF)
- JRuby
- Grails
- Groovy
- Tapestry
- ColdFusion

## Learn More

[Collapse all](#) [Expand all](#) [Collapse all](#)

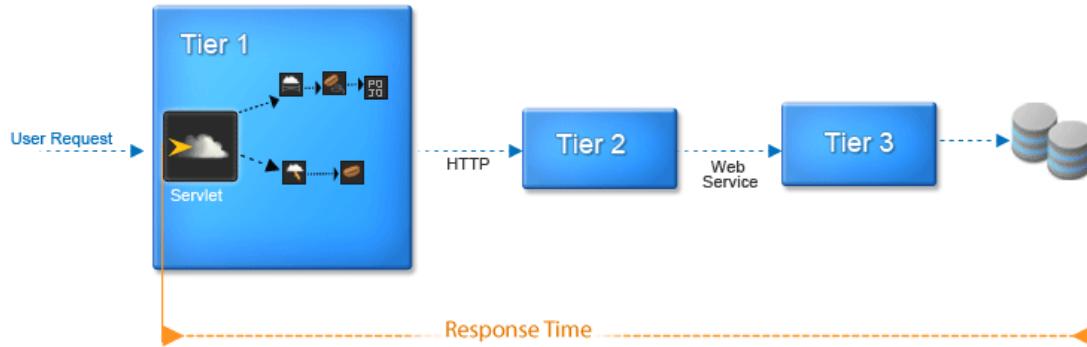
## Servlet Entry Points

- [Servlet-Based Business Transactions](#)
  - Default Naming for Servlet-Based Transactions
  - Naming Using Other URI Patterns
  - Using Headers, Cookies, and Other Parts of HTTP Requests for identifying Transactions
  - Naming Servlet Based Transactions when Different Web Contexts Require Different Naming Strategies
  - Custom Match Rules for Servlet Based Transactions
    - Example 1: URL is <http://acmeonline.com/store/checkout>
    - Example 2: URL is <http://acmeonline.com/secure/internal/updateinventory>
    - Example 3: URL is <http://acmeonline.com/orders/process?type=creditcard>
  - Using Custom Expressions in Custom Match Rules
- Request Attributes in the Custom Expression
- [Configuring Transaction Identification for REST-Style URLs](#)
- [Configuring Transaction Identification Based on Information in the Body of the Servlet Request](#)

This describes how to configure transaction entry points for Servlet-based methods that may or may not be used as part of an application framework.

## Servlet-Based Business Transactions

AppDynamics allows you to configure a transaction entry point on the invocation of the service method of a Servlet. The response time for the Servlet transaction is measured when the Servlet entry point is invoked.



### Default Naming for Servlet-Based Transactions

By default, AppDynamics identifies all Servlet-based transactions using the first two segments of the URI.

For example, if the URI for a checkout operation in an online store is

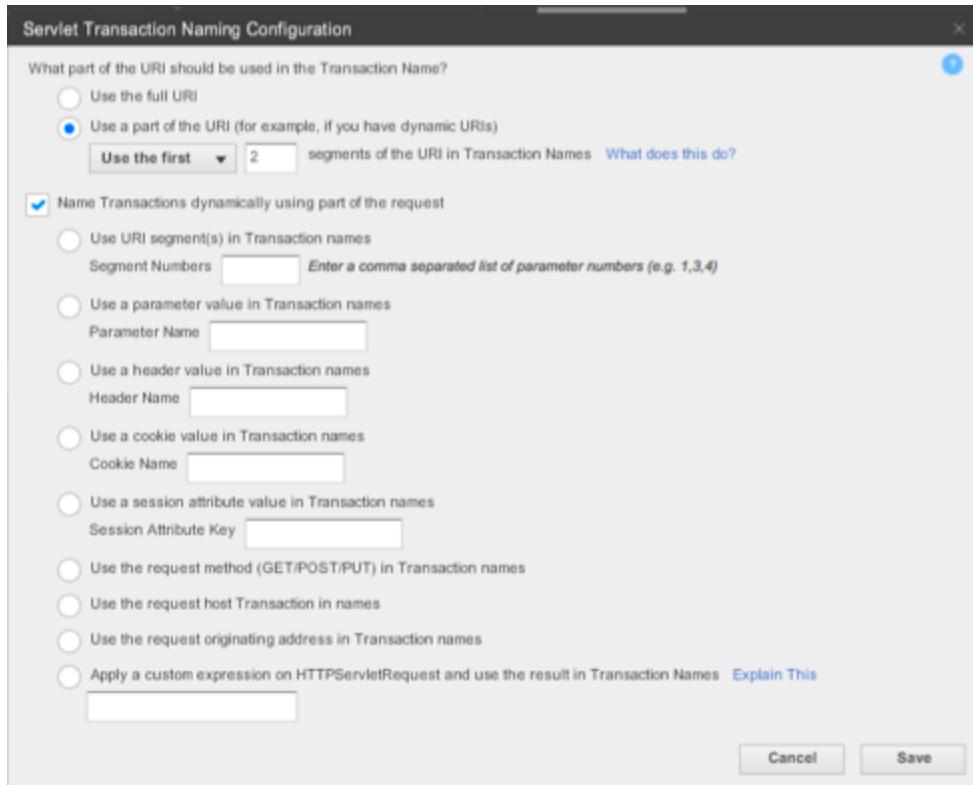
`http://acmeonline.com/store/checkout`

AppDynamics automatically names this transaction "store/checkout".

If the URI for a transfer funds operation in an online bank is

`http://acmebank.com/account/transferfunds/northerncalifornia`

AppDynamics automatically names this transaction "/account/transferfunds".



## Naming Using Other URI Patterns

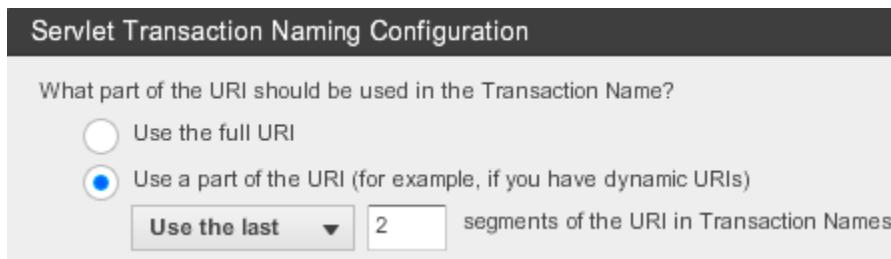
The AppDynamics default naming convention might not be optimal for all applications. You can configure AppDynamics to use different segments of the URI or other values (such as a header value, a cookie the request method, etc.).

For example the following URL represents checkout operation in ACME Online:

`http://acmeonline.com/web/store/checkout`

AppDynamics' default naming scheme identifies these transactions by the first two segments of the URI: "/web/store".

This naming convention does not indicate the functionality of the operation (checkout, add to cart, etc.) A better approach would identify such URLs using the last two segments: "store/checkout".



In certain situations, the URI might not contain enough information to be able to name the transaction. This is very common in dispatcher-style Servlet patterns where the Servlet dispatches requests based on a query parameter.

For example, for the following URL

`http://acmeonline.com/dispatcher?action=checkout`

it is ideal to name the transaction based on the value of the action parameter.

To configure this, specify the transaction detection based on the parameter value for the "action" parameter.

### Servlet Transaction Naming Configuration

What part of the URI should be used in the Transaction Name:

Use the full URI  
 Use a part of the URI (for example, if you have dynamic segments of the URL)

Use the first  2 segments of the URL

Name Transactions dynamically using part of the request

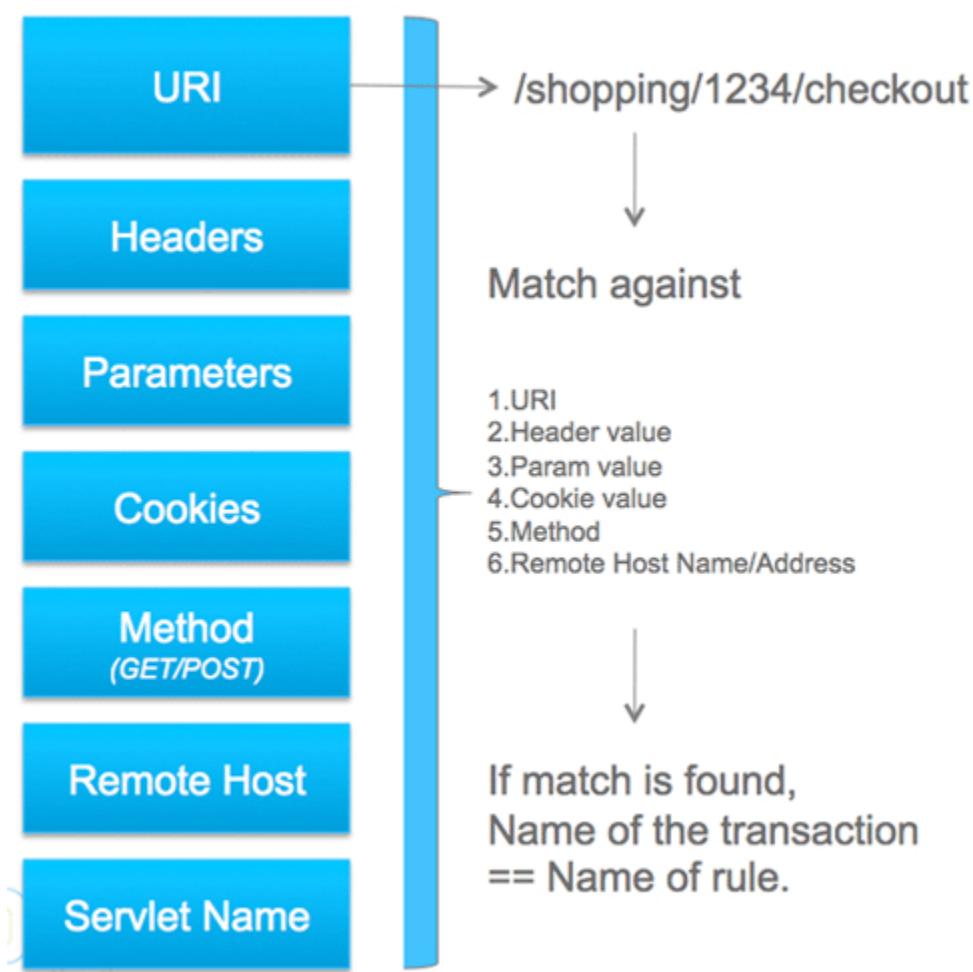
Use URI segment(s) in Transaction names  
 Segment Numbers  Enter a comma separated list of segment numbers

Use a parameter value in Transaction names  
 Parameter Name

Discovered transactions will be automatically named as "Checkout".

### Using Headers, Cookies, and Other Parts of HTTP Requests for identifying Transactions

You can also configure the naming for your Servlet based transactions using headers, cookies, and other parts of HTTP requests.



To identify all your Servlet based transactions using particular parts of the HTTP request., use the Name Transactions dynamically using part of the request option:

**Servlet Transaction Naming Configuration**

What part of the URI should be used in the Transaction Name? [?](#)

Use the full URI  
 Use a part of the URI (for example, if you have dynamic segments in the URL)

Use the first [▼](#) 2 segments of the URL

**Enable this option to name the transactions using part of the request.**

Name Transactions dynamically using part of the request

Use URI segment(s) in Transaction names  
 Segment Numbers  Enter a comma separated list of parameter numbers (e.g. 1,3,4)

Use a parameter value in Transaction names  
 Parameter Name

Use a header value in Transaction names  
 Header Name

Use a cookie value in Transaction names  
 Cookie Name

Use a session attribute value in Transaction names  
 Session Attribute Key

Use the request method (GET/POST/PUT) in Transaction names

Use the request host Transaction in names

Use the request originating address in Transaction names

Apply a custom expression on HttpServletRequest and use the result in Transaction Names

[Cancel](#) [Save](#)

Alternatively, you can also configure detection for only specific transactions using a custom match rule.

### Naming Servlet Based Transactions when Different Web Contexts Require Different Naming Strategies

In certain situations, it is ideal to modify the existing naming (which is based on the URI patterns) and name your transactions based on different web contexts. You can also configure the transaction naming to directly map the WAR files deployed on the same tier.

For example, ACME Online's entry point tier has multiple contexts deployed on a single JVM, as listed in the table below. These web contexts represent two different parts of the same business application and therefore require following different naming strategies.

Web context	Ideal naming strategy
http://acmeonline.com/store/checkout	Should use first two segments to name these requests as: /store/checkout
http://acmeonline.com/secure/internal/updateinventory	Should use last two segments to name these requests as: /internal/updateinventory

<http://acmeonline.com/orders/process?type=creditcard>

Should use the combination of parameter value for "type" and the last two segments to name such requests as: **/orders/process.creditcard**  
Such a naming strategy will ensure that the credit card orders are differentiated from other orders.

To learn more about how to configure transaction identification for these situations see [Custom match rules for Servlet based requests](#).

## Custom Match Rules for Servlet Based Transactions

To handle situations as discussed in the previous section, use custom match rules for each of the web contexts or URIs. Custom match rules let you create mutually exclusive transaction names for specific contexts.

See [Custom Match Rules](#) for general information about accessing custom match rule configuration. To configure a rule for Servlets, select **Servlet** the drop-down list.

Here are some sample rules for different URLs.

### Example 1: URL is <http://acmeonline.com/store/checkout>

New Business Transaction Match Rule - Servlet

Name: ACME

Enabled:

Priority: 0

1 Transaction Match Criteria      Split Transactions Us

Method:       URI: Starts With: /store

2      3

4  Split Transactions using request data

5  Use the first 2 segments in Transaction names

This custom rule will name all the qualifying requests as "ACME.store.checkout" transaction.

### Example 2: URL is <http://acmeonline.com/secure/internal/updateinventory>

**New Business Transaction Match Rule - Servlet**

Name: ACME  
Enabled:   
Priority: 0

**1 Transaction Match Criteria**

Method:

**2**  URI: Starts With: /secure

**3** Split Transactions Using Request Data

**4**  Split Transactions using request data

Use the first  segments in Transaction names

**5**  Use the last  segments in Transaction names

This custom rule will name all the qualifying requests as "ACME.internal.updateinventory" transaction.

**Example 3:** URL is <http://acmeonline.com/orders/process?type=creditcard>

**New Business Transaction Match Rule - Servlet**

Name: ACME  
Enabled:   
Priority: 0

**1 Transaction Match Criteria**

Method:

**2**  URI: Starts With: /orders

**3**  HTTP Parameter  
(Both GET query parameters and POST parameters can be used)

Check for parameter value: type  
Parameter Name: type  
Value: Equals creditcard

**4** Split Transactions Using Request Data

**5**  Split Transactions using request data

Use the first  segments in Transaction names

**6**  Use the last  segments in Transaction names

This custom rule will name all the qualifying requests as "ACME.orders.process.creditcard" transaction.

### Using Custom Expressions in Custom Match Rules

You can create custom expressions to name transactions based on the evaluation of a custom expression on the HttpServletRequestObject.

Suppose you want to monitor the HttpServletRequest request variable to identify the names and values of all the request parameters

passed with the request.

The following example shows a custom rule configuration based on the expression:

```
 ${getParameter(myParam)}-${getUserPrincipal().getName()}
```

which evaluates to

```
request.getParameter("myParam")+"-"+request.getUserPrincipal()
```



You can create a custom expression on the HttpServletRequest to identify all Servlet based requests (modify global discovery at the transaction naming level) or for a specific set of requests (custom rule).

## Request Attributes in the Custom Expression

A custom expression can have a combination of any of the following getter chains on the request attributes:

Getters on Request Attributes	Transaction Identification
getAuthType()	Use this option to monitor secure (or insecure) communications.
getContextPath()	Use this option to identify the user requests based on the portion of the URI.
getHeader()	Identify the requests based on request headers.
getMethod()	Identify user requests invoked by a particular method.
getPathInfo()	Identify user requests based on the extra path information associated with the URL (sent by the client when the request was made).
getQueryString()	Identify the requests based on the query string contained in the request URL after the path.
getRemoteUser()	Identify the user requests based on the login of the user making this request.
getRequestedSessionId()	Identify user requests based on the session id specified by the client.
getUserPrincipal()	Identify user requests based on the current authenticated user.

For example, the following custom expression can be used to name the business transactions using the combination of header and a particular parameter:

```
 ${getHeader(header1)}-${getParameter(myParam)}.
```

The identified transaction will be named based on the result of following expression in your application code:

```
 request.getHeader("header1")+"-"+request.getParameter("myParam")
```

## Configuring Transaction Identification for REST-Style URLs

REST applications typically have dynamic URLs where the application semantics are a part of the URL.

For example, suppose customer id or the order id can be a part of the URL. The following URL represents the checkout transaction invoked by a customer with id 1234: `http://acmeonline.com/store/cust1234/checkout`. The default discovery mechanism names this transaction "/store/cust1234". Ideally, all the customers performing a checkout operation should also be part of the checkout business transaction. Therefore, a better approach would be to name this transaction "/store/checkout".

To configure the transaction to be named "/store/checkout", use the first segment of the URL and split that URL using the third segment, thus avoiding the second (dynamic) segment.

### Configuration for global discovery

The following screenshot illustrates this configuration for global discovery.

Servlet Transaction Naming Configuration

What part of the URI should be used in the Transaction Name?

Use the full URI  
 Use a part of the URI (for example, if you have dynamic URIs)  
     1 segments of the URI in Transaction Names

Name Transactions dynamically using part of the request  
 Use URI segment(s) in Transaction names  
Segment Numbers 3 Enter a comma separated list of parameter numbers

This configuration modifies the default global discovery for all Servlet based transactions. The requests will be named as "Store.checkout" transaction.

### Configuration for custom match rule:

The following screenshot illustrates this configuration using a custom match rule.

New Business Transaction Match Rule - Servlet

Name ACME  
Enabled   
Priority 0

Transaction Match Criteria

Method GET  
 URI Equals /store

Split Transactions Using Request Data

Split Transactions using request data  
 Use the first \_\_\_\_\_ segments in Transaction names  
 Use the last \_\_\_\_\_ segments in Transaction names  
 Use URI segment(s) in Transaction names  
Segment Numbers 1,3 Enter a comma separated list of parameter numbers

Another example illustrates the rules for the following URL: `http://acmeonline.com/user/foo@bar.com/profile/profile2345/edit`. The ideal detection strategy should avoid multiple segments (in this case, we need segments 1,3, and 5).

### Configuration for global discovery:

The following screenshot illustrates the configuration for global discovery.

**Servlet Transaction Naming Configuration**

What part of the URI should be used in the Transaction Name?

Use the full URI  
 Use a part of the URI (for example, if you have dynamic URIs)  
 Use the first  1 segments of the URI in Transaction names

Name Transactions dynamically using part of the request  
 Use URI segment(s) in Transaction names  
 Segment Numbers  Enter a comma separated list of parameter numbers

This configuration modifies the default global discovery for all Servlet based transactions. The requests will be named as "User.profile.edit" transaction.

#### Configuration for custom match rule:

The following screenshot illustrates the configuration for creating a custom match rule.

**New Business Transaction Match Rule - Servlet**

Name: <input type="text" value="ACME"/>	Enabled: <input checked="" type="checkbox"/>	Priority: <input type="text" value="0"/>		
<b>Transaction Match Criteria</b> <table border="1"> <tr> <td><input type="checkbox"/> Method: <input type="button" value="GET"/></td> <td><input checked="" type="checkbox"/> URI: Equals <input type="text" value="/user"/></td> </tr> </table>			<input type="checkbox"/> Method: <input type="button" value="GET"/>	<input checked="" type="checkbox"/> URI: Equals <input type="text" value="/user"/>
<input type="checkbox"/> Method: <input type="button" value="GET"/>	<input checked="" type="checkbox"/> URI: Equals <input type="text" value="/user"/>			
<b>Split Transactions Using Request Data</b> <table border="1"> <tr> <td><input checked="" type="checkbox"/> Split Transactions using request data</td> </tr> <tr> <td> <input type="radio"/> Use the first <input type="text" value="1"/> segments in Transaction names  <input type="radio"/> Use the last <input type="text" value="1"/> segments in Transaction names  <input checked="" type="radio"/> Use URI segment(s) in Transaction names      Segment Numbers <input type="text" value="1,3,5"/> Enter a comma separated list of parameter numbers       </td> </tr> </table>			<input checked="" type="checkbox"/> Split Transactions using request data	<input type="radio"/> Use the first <input type="text" value="1"/> segments in Transaction names <input type="radio"/> Use the last <input type="text" value="1"/> segments in Transaction names <input checked="" type="radio"/> Use URI segment(s) in Transaction names Segment Numbers <input type="text" value="1,3,5"/> Enter a comma separated list of parameter numbers
<input checked="" type="checkbox"/> Split Transactions using request data				
<input type="radio"/> Use the first <input type="text" value="1"/> segments in Transaction names <input type="radio"/> Use the last <input type="text" value="1"/> segments in Transaction names <input checked="" type="radio"/> Use URI segment(s) in Transaction names Segment Numbers <input type="text" value="1,3,5"/> Enter a comma separated list of parameter numbers				

## Configuring Transaction Identification Based on Information in the Body of the Servlet Request

In certain situations, you might have an application that receives an XML/JSON payload as part of the POST request.

If the category of processing is a part of the XML/JSON, neither the URI nor the parameters/headers have enough information to name the transaction. Only the XML contents that can provide correct naming configuration.  
See [Identify Transactions Based on DOM Parsing Incoming XML Payload](#) and [Identify Transactions for Java XML Binding Frameworks](#).

## Servlet Transaction Detection Scenarios

### Identify Transactions Based on DOM Parsing Incoming XML Payload

- Business Transactions and XML Payload
  - Identifying the Business Transaction Using the XPath Expression
    - To configure a custom match rule
  - Learn More

This topic describes how to identify transactions when an XML is posted to a Servlet.

#### Business Transactions and XML Payload

The XML contains the naming information for the Business Transaction. The Servlet uses a DOM parser to parse the posted XML into a DOM object.



**Posted XML is parsed into a DOM object**

#### ***Identifying the Business Transaction Using the XPath Expression***

For example, the following XML posts an order for three items. The order uses credit card processing.

```

<acme>
  <order>
    <type>creditcard</type>
    <item>Item1</item>
    <item>Item2</item>
    <item>Item3</item>
  </order>
</acme>

```

The URL is:

```
http://acmeonline.com/store
```

The doPost method of the Servlet is:

```

public void doPost(HttpServletRequest req, HttpServletResponse resp)
{
    DocumentBuilderFactory docFactory =
    DocumentBuilderFactory.newInstance();
    DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
    Document doc = docBuilder.parse(req.getInputStream());

    Element element = doc.getDocumentElement();

    //read the type of order
    //read all the items
    processOrder(orderType,items)
    ....
}

```

The XPath expression "//order/type" on this XML payload evaluates to "creditcard".

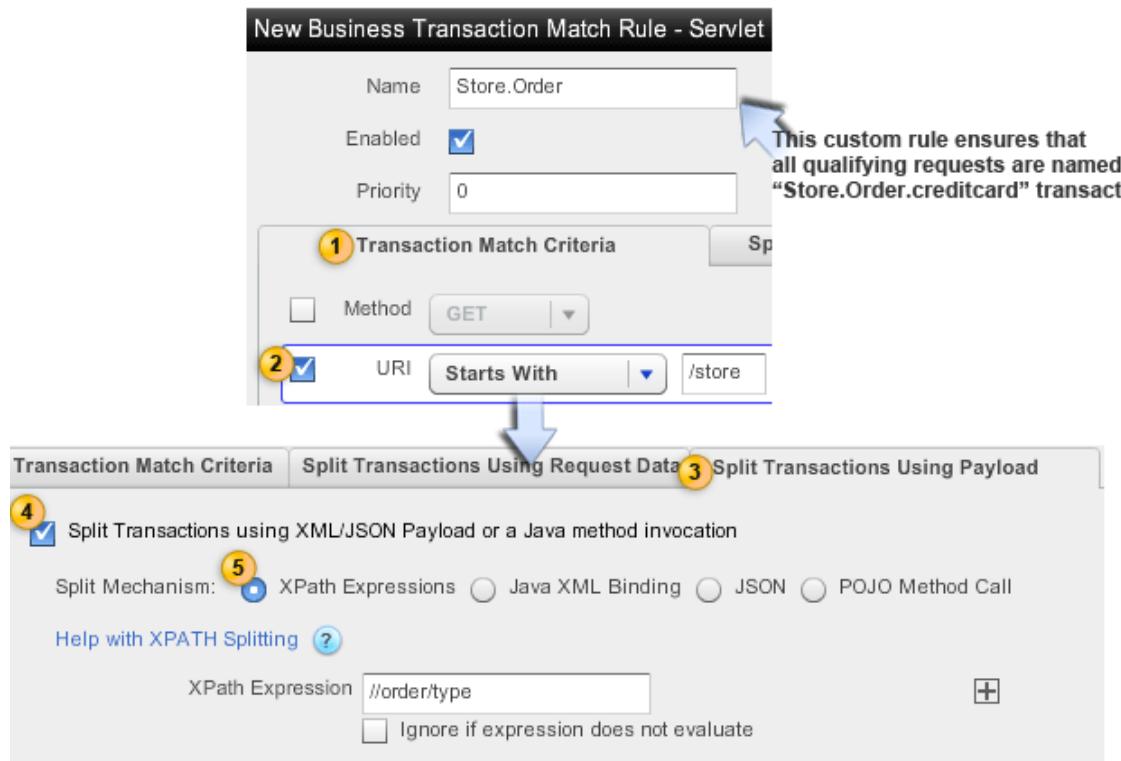
This value correctly identifies the type of the order and therefore should be used to name the "Order" transaction.

To identify the Business Transactions in this manner, first configure a custom match rule that automatically intercepts the method that parses the XML and gets the DOM object.

You use the XPath expression in the custom rule so that it names the transaction, for example "Store.order.creditcard". Though the name is not obtained until the XML is parsed, AppDynamics measures the duration of the business transaction to include the execution of the doPost() method.

#### To configure a custom match rule

1. Navigate to the custom rule section for Servlets.
2. In the **Transaction Match Criteria** tab, specify the URI.
3. In the **Split Transactions Using Payloads** tab, enable **Split transactions using XML/JSON Payload or a Java method invocation**.
4. Set the split mechanism to **XPath Expressions**.
5. Enter the XPath expression that you want to set as the Entry Point. The result of the XPath expression will be appended to the name of the Business Transaction.



You can use one or more XPath expressions to chain the names generated for the Business Transaction.

If the expression does not evaluate to a value, the transaction will not be identified.

#### Learn More

- [Servlet Entry Points](#)

### Identify Transactions Based on POJO Method Invoked By a Servlet

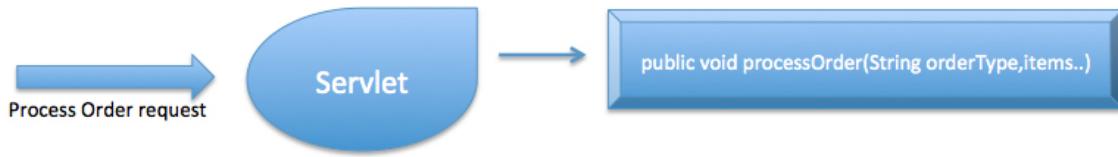
- [Using a Java Method to Name a Transaction](#)
  - To configure the custom match rule
- [Learn More](#)

### Using a Java Method to Name a Transaction

You can use a Java method to name the transaction where:

- You might not have a clear URI pattern or
- You are using XML/JSON frameworks that are currently not supported by AppDynamics.

The following illustration shows how the Servlet that invokes the POJO method holds the transaction name.



For example, consider the `processOrder()` method. The order is parsed using any type of method and eventually when the `processOrder()` method is invoked, a correct approach to name transaction is to capture the first parameter to the `processOrder()` method.

The following URL is used by these requests: <http://acmeonline.com/store>. Following code snippet shows the `doPost()` method of the Servlet:

```
public void doPost(HttpServletRequest req, HttpServletResponse resp)
{
    //process the data from the sevlet request and get the orderType and the items
    processOrder(orderType,item)
    ....
}
public void processOrder(String orderType, String item)
{
    //process order
}
```

The `processOrder()` method has the information which can correctly derive the type of the order and also the name of the transaction. To identify these requests as a single transaction, configure a custom match rule.

#### To configure the custom match rule

1. Go to the custom rule section for Servlet Entry Points
2. In the **Transaction Match Criteria** tab, specify the URI.
3. In the **Split Transactions Using Payload** tab, enable Split transactions using XML/JSON Payload or a Java method invocation.
4. Select POJO Method Call as the split mechanism .
5. Enter the class and the method name.
6. If the method is overloaded, also specify the details for the arguments.
7. Optionally, you can name your transactions by defining multiple methods in a getter chain in the Method Call Chain field.

The following screenshot displays the configuration of a custom match rule which will name all the qualifying requests into a "Store.order.creditcard" transaction:

**New Business Transaction Match Rule - Servlet**

Name: Store.Order  
Enabled:   
Priority: 0

**1 Transaction Match Criteria**

Method: GET  
URI: Equals /store

**2 Split Transactions Using Request Data**

**3 Split Transactions Using Payload**

**4 Split Transactions using XML/JSON Payload or a Java method invocation**

Split Mechanism:  XPath Expressions  Java XML Binding  JSON  POJO Method Call

Help with POJO Splitting [?](#)

Class Name: com.acme.Order.ProcessOrder  
Method Name: processOrder()  
Return Type:   
Number of Arguments: 2  
Argument Index: 0  
Method Call Chain:  +  
*For example: getPerson().getAddress().getStreet()*

This custom rule ensures that the processOrder method is automatically intercepted.

Although the name is not obtained till the processOrder() method is called, the time for the transaction will include all of the doGet() method.

In addition to the parameter, you can also specify either the return type or a recursive getter chain on the object to name the transaction. For example, if the method parameter points to a complex object like PurchaseOrder, you can use something like getPurchaseDetails().getType() to correctly name the transaction.

#### Learn More

- [Servlet Entry Points](#)

#### Identify Transactions for Java XML Binding Frameworks

- To Configure the Custom Match Rule
- Supported Java XML data binding frameworks
- [Learn More](#)

The following illustration shows the situation in which an XML is posted to a Servlet. The Servlet uses an XML-to-Java binding framework, such as XMLBeans or Castor, to unmarshal and read the posted XML payload.



The XML payload contains the naming information for the transactions.

In the following example, an XML payload posts an order for three items. It uses a credit card to process the order.

The URL is: <http://acmeonline.com/store>

```

<acme>
  <order>
    <type>creditcard</type>
    <item>Item1</item>
    <item>Item2</item>
    <item>Item3</item>
  </order>
</acme>

```

The following code snippet shows the doPost() method of the Servlet:

```

public void doPost(HttpServletRequest req, HttpServletResponse resp)
{
    PurchaseOrderDocument poDoc = PurchaseOrderDocument.Factory.parse(po);

    PurchaseOrder po = poDoc.getPurchaseOrder();
    \\
    String orderType = po.getOrderType();

    //read all the items
    processOrder(orderType,items)

    ...
}

```

After the posted XML is unmarshalled to the PurchaseOrder data object, the getOrderType() method should be used to identify the type of the order.

#### To Configure the Custom Match Rule

1. Navigate to the custom rule section for **Servlet Entry Points**.
2. In the **Transaction Match Criteria** tab, specify the URI.
3. In the **Split Transactions Using Payload** tab, check Split transactions using XML/JSON Payload or a Java method invocation.
4. Select Java XML Binding as the split mechanism.
5. Enter the class name and the method name.

The screenshot below shows a custom match rule which identifies the business transaction for this example as "Store.order.creditcard":

**New Business Transaction Match Rule - Servlet**

This custom rule ensures that all qualifying requests are named as "Store.Order.creditcard" transaction.

Name	Store.Order
Enabled	<input checked="" type="checkbox"/>
Priority	0

**1 Transaction Match Criteria**

Method: GET

**2 URI Equals /store**

**3 Split Transactions Using Request Data**

**4 Split Transactions using XML/JSON Payload or a Java method invocation**

Split Mechanism:  XPath Expressions  Java XML Binding  JSON  POJO Method Call

[Help with Java XML Binding Splitting](#)

Unmarshaled Class Name: PurchaseOrderDocument

Method Call Chain: getPurchaseOrder() [+](#)

For example: `getPerson().getAddress().getStreet()`

This custom rule ensures that the method in XMLBeans (which unmarshals XML to Java objects) is automatically intercepted. It also ensures that the `getOrderType()` method is applied on the Java data object only if it is the `PurchaseOrder` data object.

If the name of the transaction is not on a first level getter on the unmarshalled object, you can also use a recursive getter chain such as `getOrderType().getOrder()` to get the name.

Although the transaction name is not obtained until the XML is unmarshalled, the response time for the transaction is calculated from the `doGet()` method.

## Supported Java XML data binding frameworks

The following Java XML data binding frameworks are supported:

- Castor
- JAXB
- JibX
- XMLBeans
- XStream

## Learn More

- [Servlet Entry Points](#)

## Identify Transactions Based on JSON Payload

- [Example Configuration for a JSON Payload](#)
- [Learn More](#)

## Example Configuration for a JSON Payload

The following illustration shows a JSON payload posted to a Servlet and the Servlet unmarshalls the payload.



The JSON contains the naming information for the transactions.

For example, the following JSON payload posts an "order" for an item "car" and uses creditcard for processing the order. The URL is:

\*<http://acmeonline.com/store>\*

```

order
: {
type:creditcard,
id:123,
name:Car,
price:23
}}

```

The following code snippet shows the doPost method of the Servlet:

```

public void doPost(HttpServletRequest req, HttpServletResponse resp)
{
    //create JSONObject from servlet input stream
    String orderType = jsonObject.get("type")\\
    //read the item for the order\\
    processOrder(orderType,item)\\
    ....\\
}

```

After the posted JSON payload is unmarshalled to the JSON object, the "type" key is required to identify the type of the order. In this case, this key uniquely identifies the business transaction.

To configure this rule:

1. Go to the custom rule section for [Servlet Entry Points](#).
2. Under "Transaction Match Criteria" specify the URI.
3. Under "Split Transactions Using Payloads", enable "Split transactions using XML/JSON Payload or a Java method invocation".
4. Select the split mechanism as "JSON".
5. Enter the JSON object key.

The following screenshot displays the configuration for a custom match rule that will name all the qualifying requests into a single transaction called "Store.Order.creditcard".

This custom rule ensures that all qualifying requests are named as "Store.Order.creditcard" transaction.

**1** Transaction Match Criteria      **2** URI Equals /store

**3** Split Transactions Using Payload

**4**  Split Transactions using XML/JSON Payload or a Java method invocation

Split Mechanism:  XPath Expressions  Java XML Binding **5**  JSON  POJO Method Call

NOTE: The 'enable-json-bci-rules' agent property must be set to true for each node in this Tier for this match rule to work.

Help with JSON Splitting ?

JSON Object Key type

6. Set the property "enable-json-bci-rules" to "true" for each node to enable this custom rule. See [App Agent Node Properties](#).

This configuration ensures that the JSONObject and the get("\$JSON\_Object\_Key") method on this object are intercepted automatically to get the name of the transaction. Although the transaction name is not obtained until the JSON object is unmarshalled, the response time for the transaction will be calculated from the doGet method.

#### Learn More

- [Servlet Entry Points](#)
- [App Agent Node Properties](#)

## Struts Entry Points

- Struts-based Transactions
- Struts Request Names
- Custom Match Rules for Struts Transactions
- Exclude Rules for Struts Actions or Methods

## Struts-based Transactions

When your application uses Struts to service user requests, AppDynamics intercepts individual Struts Action invocations and names the user requests based on the Struts action names. A Struts entry point is a Struts Action that is being invoked.

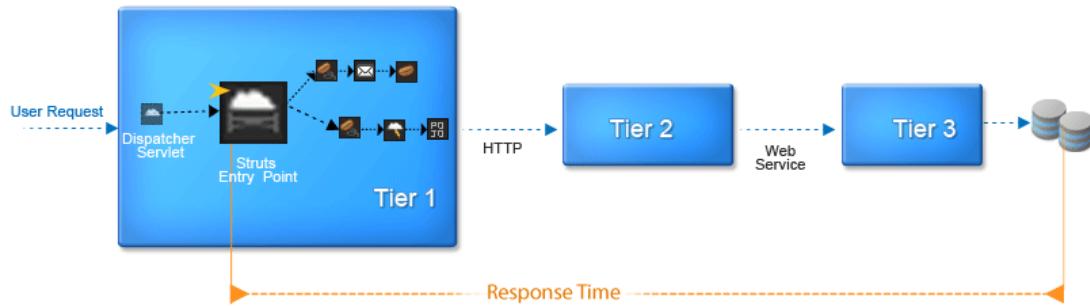
AppDynamics supports following versions of Struts:

- Struts 1.x
- Struts 2.x

Struts Action invocations are typically preceded by a dispatcher Servlet, but identification is deferred to the Struts Action. This ensures that the user requests are identified based on the Struts Action and not from the generic URL for Dispatcher Servlet.

The response time for the Struts-based transaction is measured when the Struts entry point is invoked.

The following figure shows the identification process for Struts Action entry points.



## Struts Request Names

When a Struts Action is invoked, by default AppDynamics identifies the request using the name of Struts Action and the name of the method. All automatically discovered Struts-based transactions are thus named using the convention <Action Name>. <Method Name>.

For example, if an action called ViewCart is invoked with the SendItems(), the transaction is named "ViewCart.SendItems".

For Struts 1.x the method name is always "execute".

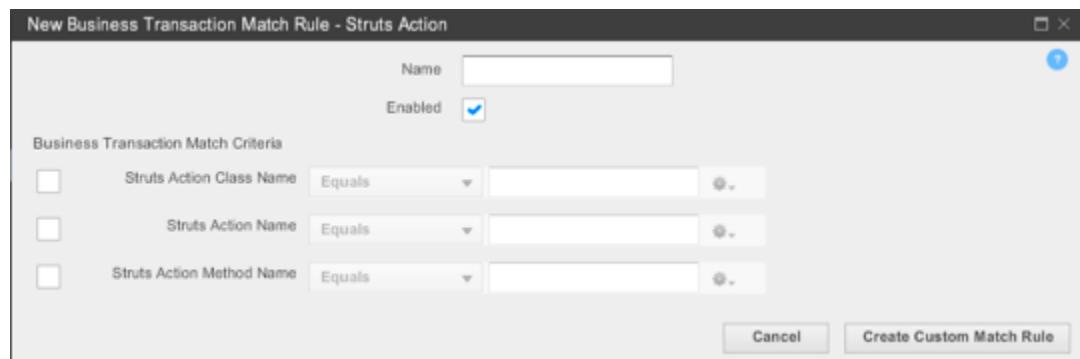
You can rename or exclude auto-discovered transactions. See [Business Transaction List Operations](#).

## Custom Match Rules for Struts Transactions

For finer control over the naming of Struts-based transactions, use custom match rules.

A custom match rule lets you specify customized names for your Struts-based requests. You can also group multiple Struts invocations into a single business transaction using custom match rules. See [Custom Match Rules](#) for information about accessing the configuration screens.

The matching criteria for creating the rule are: Struts Action class names, Struts Action names, and Struts Action method names.



## Exclude Rules for Struts Actions or Methods

To prevent specific Struts Actions and methods from being monitored add an exclude rule. See [Exclude Rules](#). The criteria for Struts exclude rules are the same as those for custom match rules.

## Web Service Entry Points

- Web Services-based Transactions
  - Default Naming
  - Grouping Web Service Actions or Operation Names into a Business Transaction
  - Exclude Rules

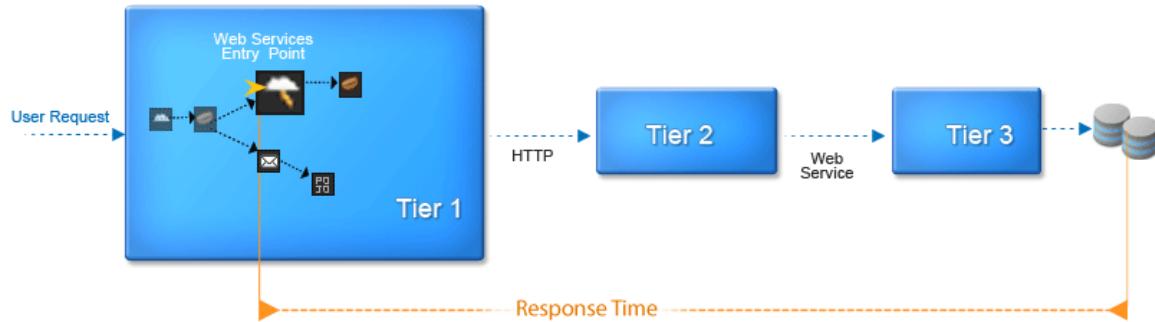
This topic discusses Web Service Entry Points.

## Web Services-based Transactions

When your application uses Web Services to service user requests, AppDynamics intercepts the Web Service invocations and names requests based on the Web Service action names and operation name. A Web Service entry point is a Web Service end point that is being invoked.

This is relevant only when the Web Service invocation is part of the entry point tier and not in a downstream tier.

Web Service invocations are usually preceded by a dispatcher Servlet, but identification is deferred to the Web Service endpoints. This configuration ensures that the requests are identified based on the Web Service and not based on the generic URL for the dispatcher Servlet.



### Default Naming

When the Web Service end point is invoked, the request is named after the Web Service name and the operation name.

For example, if a service called CartService is invoked with the Checkout operation, it is named "CartService.Checkout".

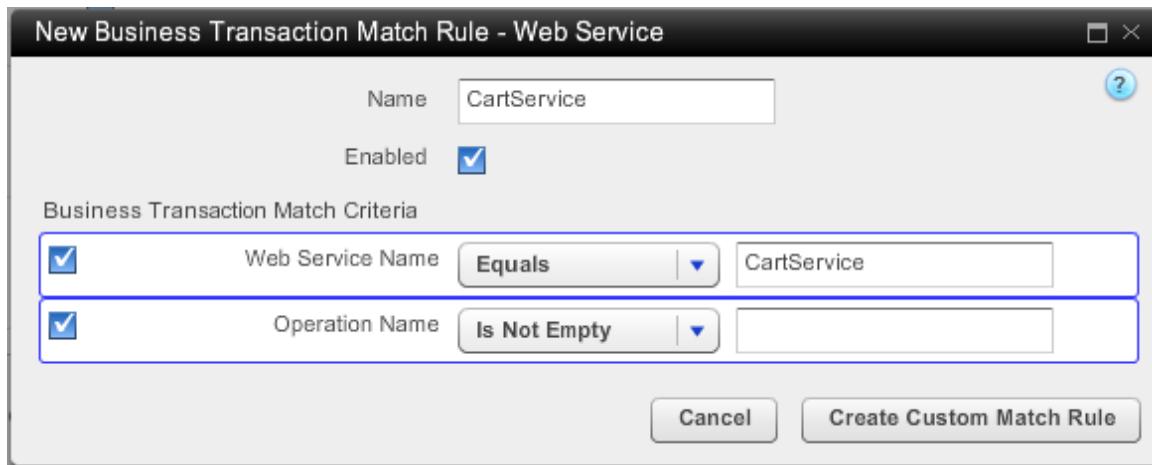
You can rename or exclude these automatically discovered transactions. See [Business Transaction List Operations](#).

### Grouping Web Service Actions or Operation Names into a Business Transaction

If you want finer control over naming of Web Service requests, you can create custom match rules for Web Services. See [Custom Match Rules](#) for information about accessing the configuration screens.

The matching criteria for Web Service rules are Web Service Name and Operation Name

The following example names all operations for the Web Service named "CartService":



## Exclude Rules

To exclude specific Web Services or operation names from detection, add an exclude rule. See [Exclude Rules](#). The criteria for Web Service exclude rules are the same as those for custom match rules.

## POJO Entry Points

- Default Identification of POJO Transactions
- Recommended Practices for Defining a POJO Entry Point
- Custom Match Rules for POJO Transactions
  - POJO Transaction as a Background Task
  - Names for Custom POJO-Based Transactions
- Splitting POJO-Based Transactions
  - To configure transaction splitting
  - Using Method parameter for dynamically naming the transactions
- Exclude Rules for POJO Transactions

Not all business processing can be implemented using Web entry points for popular frameworks. Your application may perform batch processing in all types of containers. You may be using a framework that AppDynamics does not automatically detect. Or maybe you are using pure Java.

In these situations, to enable detection of your business transaction, configure a custom match rule for a POJO (Plain Old Java Object) entry point. The rule should be defined on the class/method that is the most appropriate entry point. Someone who is familiar with your application code should help make this determination. See [Recommended Practices for Defining a POJO Entry Point](#).

AppDynamics measures performance data for POJO transactions as for any other transactions. The response time for the transaction is measured from the POJO entry point, and the remote calls are tracked the same way as remote calls for a Servlet's Service method.

## Default Identification of POJO Transactions

By default, the AppDynamics discovery for POJO-based requests captures either a single or a very small group of Servlet URLs. AppDynamics identifies the POJO-based transactions automatically using the class name and method name convention: <ClassName>. <MethodName>. These transactions are displayed on the Business Transactions List.

In certain cases the POJO transactions are not identified separately either because the POJO does not implement a predefined interface or because it does not have a predefined annotation. Auto-discovery for these transactions is not possible.

In some circumstances, AppDynamics' default discovery rules for Servlets take precedence over the POJO-based transaction. For example, the ACME Online application has a Servlet-handled checkout operation with URI pattern: /cart. The Servlet handling the operation reads the HTTP POST parameters to determine if this is a checkout operation. With this type of user request, the Servlet dispatcher parses the request data to determine the requested operation and then dispatches the request to the POJO for handling.

## Recommended Practices for Defining a POJO Entry Point

The POJO entry point is the Java method that starts the transaction.

The most important consideration in defining a POJO entry point is to choose a method that begins and ends every time the specific business transaction is invoked.

For example, consider the method execution sequence:

```
com.foo.threadpool.WorkerThread.run()
    calls com.foo.threadpool.WorkerThread.runInternal()
        calls com.foo.Job.run()
```

The first two calls to run() method are the blocking methods that accept a job and invoke it.

The Job.run() method is the actual unit of work, because Job is executed every time the business transaction is invoked and finishes when the business transaction finishes.

Methods like these are the best candidates for POJO entry points.

## Custom Match Rules for POJO Transactions

If you are not getting the required visibility with auto-discovered transactions, create a custom match rule for a POJO transaction. See [Custom Match Rules](#).

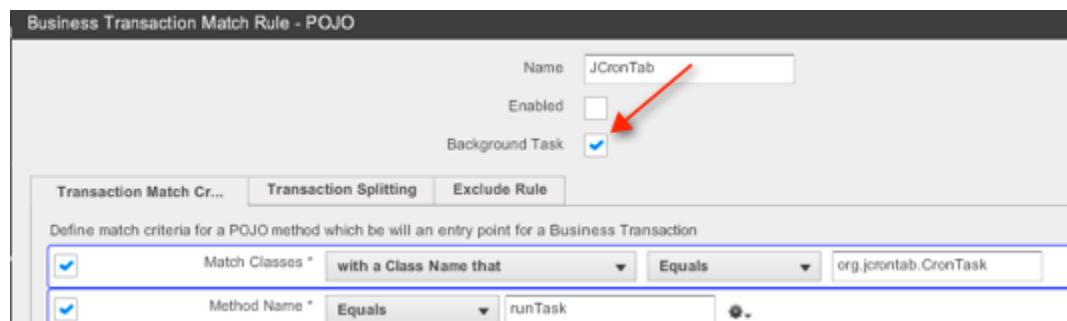
Creating a POJO transaction at a minimum involves setting the entry point.

You may optionally refine the naming of a POJO-based transaction by transaction splitting.

If you are running on IBM JVM v1.5 or v1.6, you must restart the JVM after defining the custom rules.

## POJO Transaction as a Background Task

You can specify that a POJO request run as a background task by checking the Background Task check box in the configuration.



When a request runs as a background task, AppDynamics reports only Business Transaction metrics for the request. It does not aggregate response time and calls metrics at the tier and application levels for background tasks. This is ensures that background tasks do not distort the baselines for the business application. Also, you can set a separate set of thresholds for background tasks. See [Background Task Monitoring](#)

## Names for Custom POJO-Based Transactions

By default, when you create a custom match rule for a POJO-based transaction, the name of the transaction is based on the name of the custom rule.

For example, consider entry point based on the com.acme.AbstractProcessor superclass, which defines a process() method, which is extended by its child classes: SalesProcessor, InventoryProcessor, BacklogProcessor.

You can define the custom rule on the method defined in the superclass:

New Business Transaction Match Rule - POJO

Name	<input type="text" value="Process"/>	<a href="#">?</a>
Enabled	<input checked="" type="checkbox"/>	
Background Task	<input type="checkbox"/>	
<input type="button" value="Transaction Match Crit..."/> <input type="button" value="Transaction Splitting"/> <input type="button" value="Exclude Rule"/>		
Define match criteria for a POJO method which will be an entry point for a Business Transaction		
<input checked="" type="checkbox"/> Match Classes * <input type="text" value="that extends a Super Class that"/> <input type="button" value="Down"/> <input type="button" value="Equals"/> <input type="button" value="Down"/> <input type="text" value="com.acme.AbstractProcessor"/>		
<input checked="" type="checkbox"/> Method Name * <input type="button" value="Equals"/> <input type="button" value="Down"/> <input type="text" value="process"/>		
<input type="button" value="Cancel"/> <input type="button" value="Create Custom Match Rule"/>		

Similarly, you can define a custom rule matching on an interface named `com.acme.IProcessor`, which defines a `process()` method that is implemented by the `SalesProcessor`, `InventoryProcessor`, `BacklogProcessor` classes.

New Business Transaction Match Rule - POJO

Name	<input type="text"/>	<a href="#">?</a>
Enabled	<input checked="" type="checkbox"/>	
Background Task	<input type="checkbox"/>	
<input type="button" value="Transaction Match Criteria"/> <input type="button" value="Transaction Splitting"/> <input type="button" value="Exclude Rule"/>		
Define match criteria for a POJO method which will be an entry point for a Business Transaction		
<input checked="" type="checkbox"/> Match Classes * <input type="text" value="that implements an Interface which"/> <input type="button" value="Down"/> <input type="button" value="Equals"/> <input type="button" value="Down"/> <input type="text" value="com.acme.IProcessor"/>		
<input checked="" type="checkbox"/> Method Name * <input type="button" value="Equals"/> <input type="button" value="Down"/> <input type="text"/>		
<input type="button" value="Cancel"/> <input type="button" value="Create Custom Match Rule"/>		

You can also configure a custom rule based on the Annotations.

For example, if all processor classes are annotated with `@com.acme.Processor`, a custom rule should be defined using annotation.

New Business Transaction Match Rule - POJO

Name	<input type="text" value="Process"/>	<a href="#">?</a>
Enabled	<input checked="" type="checkbox"/>	
Background Task	<input type="checkbox"/>	
<input type="button" value="Transaction Match Criteria"/> <input type="button" value="Transaction Splitting"/> <input type="button" value="Exclude Rule"/>		
Define match criteria for a POJO method which will be an entry point for a Business Transaction		
<input checked="" type="checkbox"/> Match Classes * <input type="text" value="that has an Annotation which"/> <input type="button" value="Down"/> <input type="button" value="Equals"/> <input type="button" value="Down"/> <input type="text" value="com.acme.Processor"/>		
<input checked="" type="checkbox"/> Method Name * <input type="button" value="Equals"/> <input type="button" value="Down"/> <input type="text" value="process"/>		
<input type="button" value="Cancel"/> <input type="button" value="Create Custom Match Rule"/>		

By default, in these cases the business transaction started when any `process()` is invoked is named Process, based on the name of the

custom rule. To refine the transaction name to reflect the specific method called (Process.SalesProcessor, Process.InventoryProcessor, Process.BacklogProcessor) use transaction splitting.

## Splitting POJO-Based Transactions

By default, when you create a custom rule for POJO transactions, all the qualifying requests are identified by the name of the custom rule.

However, in many situations it is preferable to split POJO-based transactions, especially for nodes that execute scheduled jobs.

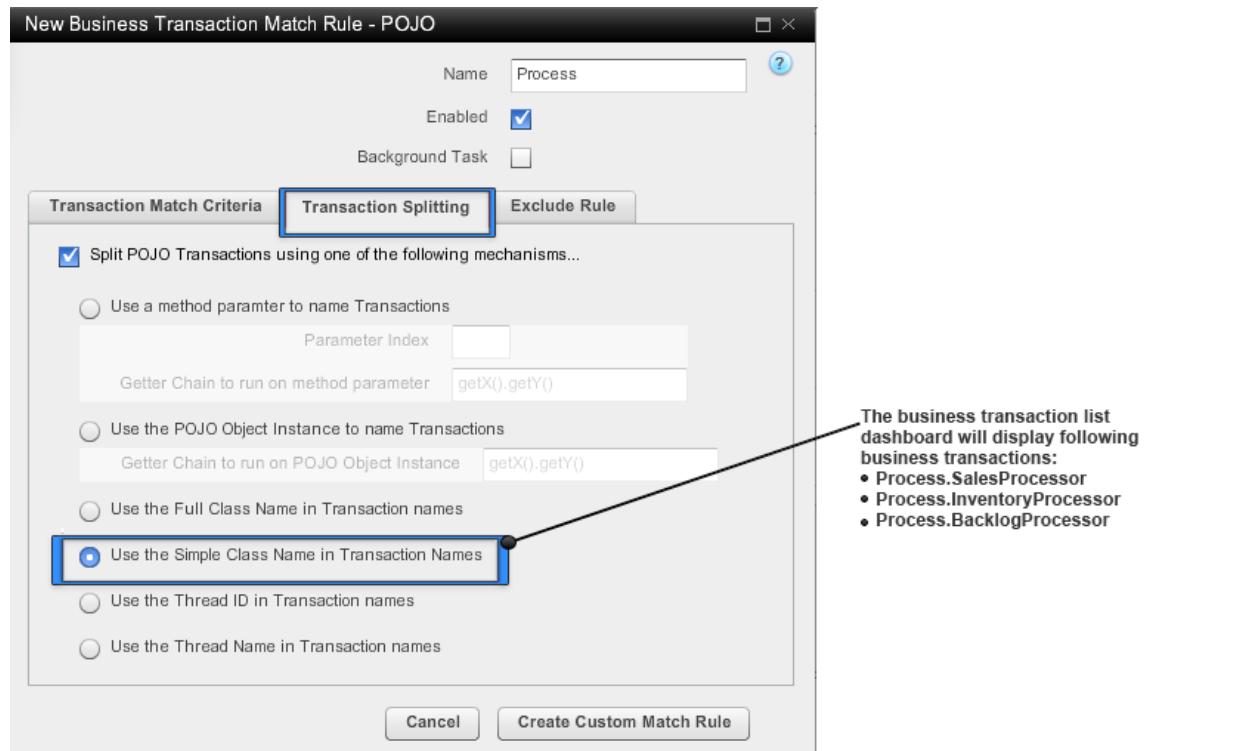
For example, if multiple classes have the same method and are instrumented using the same rule, when the method is invoked the class name of the instance being invoked can be used to classify the request.

If you split the transaction based on the simple class name, instead of one business transaction named Process, the transaction that is started when the process() method is invoked is named based on the rule name combined with the class name: either Process.SalesProcessor, Process.InventoryProcessor, or Process.BacklogProcessor.

In some cases you want to split the transaction based on the value of a parameter in the entry point method. For example, you could configure the split on the following process() method:

```
public void process(String jobType, String otherParameters...)
```

where the jobType parameter could be Sales, Inventory or Backlog.



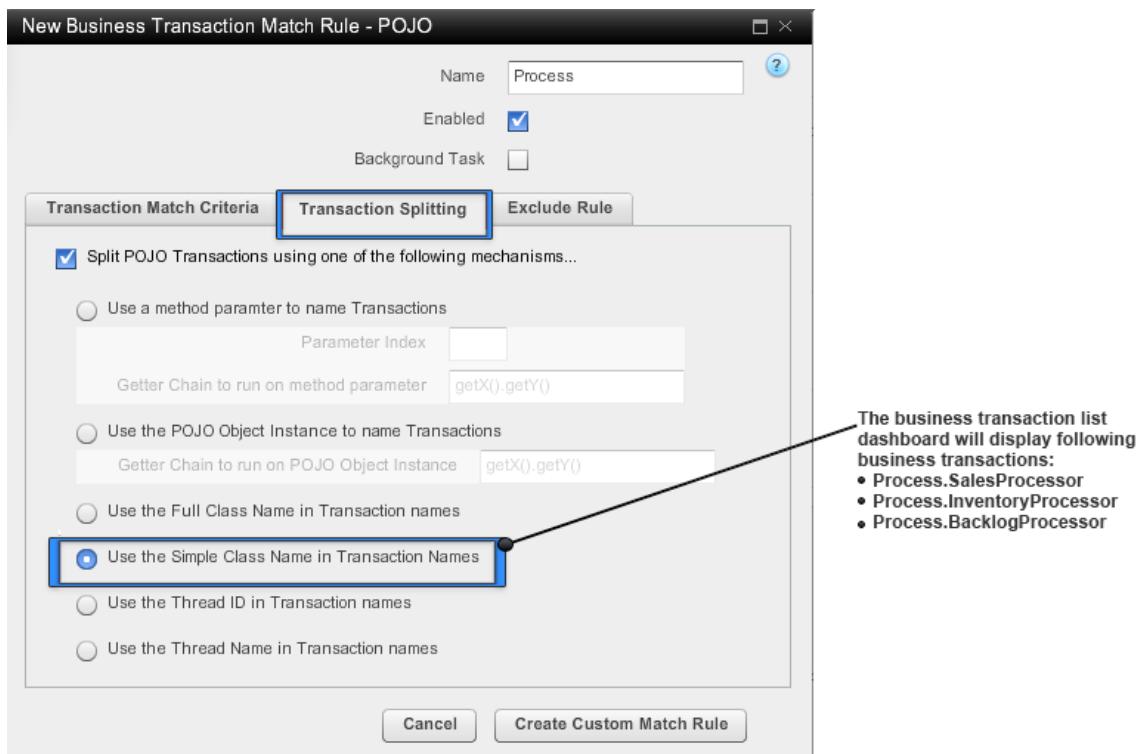
You can name POJO transactions dynamically using the following mechanisms:

- method parameter
- POJO object instance
- fully qualified class name
- simple class name
- thread ID
- thread name

In all cases, the name of the rule is prepended to the dynamically-generated name to form the business transaction name.

## To configure transaction splitting

1. In the Transaction Splitting tab of the Business Transaction Match Rule - POJO window, check the Split POJO Transactions box to enable transaction splitting.
2. Select the mechanism to use to split the transaction.  
If you are specifying a method parameter, enter the zero-based parameter index of the parameter  
If the parameter is a complex type, specify the getter chain to use used to derive the transaction name.  
If you are specifying a POJO object instance, specify the getter chain. See [Getter Chains in .NET Configurations](#).
3. Click **Save**.



## Using Method parameter for dynamically naming the transactions

Suppose in the ACME Online example, instead of the super-class or interface, the type of the processing is passed in as a parameter.

For example:

```
public void process(String jobType, String otherParameters...)
```

In this case, it would be appropriate to name the transaction based on the value of Job type. This Job type is passed as the parameter. To specify a custom rule for method parameter:

1. Specify the details for the custom rule in the **Transaction Match Criteria** tab.
2. Click the **Transaction Splitting** tab.
3. Check the Split POJO transactions using following mechanisms checkbox.
4. Select the option for method parameter.
5. Specify the details for the parameters.

You can use a getter chain if the parameter is of complex type in order to derive a string value, which can then be used for the transaction name. See [Getter Chains in .NET Configurations](#).

## Exclude Rules for POJO Transactions

To prevent configured transaction splitting from being applied in certain situations, create an exclude rule defined on the output of the transaction splitting.

The following condition will be applied to the output of the Transaction Splitting configuration. All traffic matching this Exclude Rule will be registered with a single Business Transaction with the same name as this Match Rule (splitting will not happen). This is a way to define a custom rule to split Transactions in all but certain specific conditions.

**Exclude Criteria**

<input checked="" type="checkbox"/> Transaction Splitting Output	Equals	<input type="text"/>
Equals Starts With Ends With Contains Matches Reg Ex		

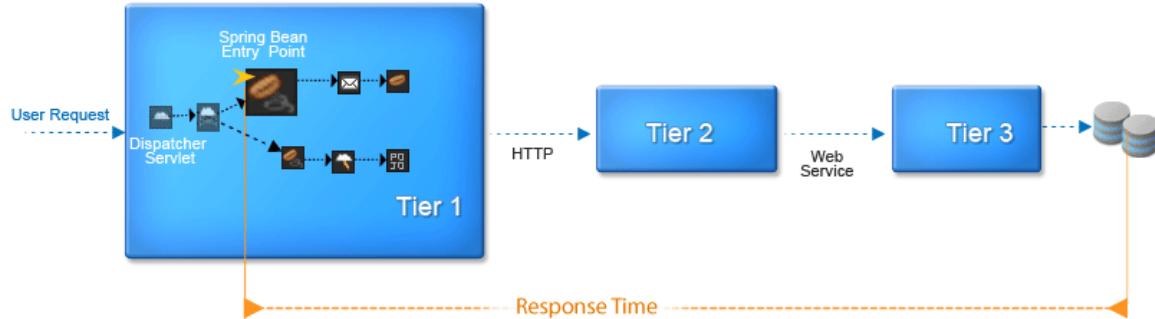
## Spring Bean Entry Points

- Spring Bean-based Transactions
- Default Naming for Spring Bean Requests
  - To Enable Auto-discovery for Spring Bean entry points
- Custom Match Rules for Spring Bean Requests
- Exclude Rules Spring Bean Transactions

This topic describes how to configure transaction entry points for Spring Bean requests.

## Spring Bean-based Transactions

AppDynamics allows you to configure a transaction entry point for a particular method for a particular bean in your environment. The response time is measured from when the Spring Bean entry point is invoked.



## Default Naming for Spring Bean Requests

When the automatic discovery for a Spring Bean based request is turned on, AppDynamics automatically identifies all the Spring Beans based transactions and names these transactions using the following format:

BeanName . MethodName

By default, the transaction discovery for Spring Bean-based requests is turned off.

### To Enable Auto-discovery for Spring Bean entry points

1. Access the transaction detection configurations screen and select the tier to configure. See [To Access Business Transaction Detection Configuration](#)
2. In the Spring Bean entry in the Entry Points section check the Automatic Transaction Detection check box.

#### ▼ Entry Points

Type	Transaction Monitoring	Description
	Automatic Transaction Detection	
Servlet	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Servlet requests <input type="checkbox"/> Enable Servlet Filter Detection
Struts Action	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Struts Action invocations Transactions will be named: ActionName.MethodName
Web Service	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Web Service requests Transactions will be named: ServiceName.OperationName
POJO	<input checked="" type="checkbox"/> Enabled	Any Java method can be the entry point for a Business Transaction. The class to which the method belongs to can be picked using different parameters like its name, its super class name, the interfaces it implements, or the annotations it has.
Spring Bean	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically for all Spring Bean invocations Transactions will be named: BeanName.MethodName
EJB	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Discover Transactions automatically for all EJB invocations Transactions will be named: EJBNName.MethodName

## Custom Match Rules for Spring Bean Requests

If you are not getting the required visibility with the auto-discovered transactions, you can create a custom match rule for a Spring Bean based transaction. See [Custom Match Rules](#).

The following example creates a custom match rule for the placeOrder method in the orderManager bean.

New Business Transaction Match Rule - Spring Bean

Name	ACME	?	
Enabled	<input checked="" type="checkbox"/>		
Business Transaction Match Criteria			
<input checked="" type="checkbox"/>	Bean ID	Contains	orderManager
<input checked="" type="checkbox"/>	Method	Equals	placeOrder
<input type="checkbox"/>	Class Name	Equals	
<input type="checkbox"/>	Extends	Equals	
<input type="checkbox"/>	Implements	Equals	
		<input type="button" value="Cancel"/>	<input type="button" value="Create Custom Match Rule"/>

This custom rule will name all the qualifying requests as "ACME.orderManager.placeOrder".

## Exclude Rules Spring Bean Transactions

To exclude specific Spring Bean transactions from detection add an exclude rule. See [Exclude Rules](#). The criteria for Spring Bean exclude rules are the same as those for custom match rules.

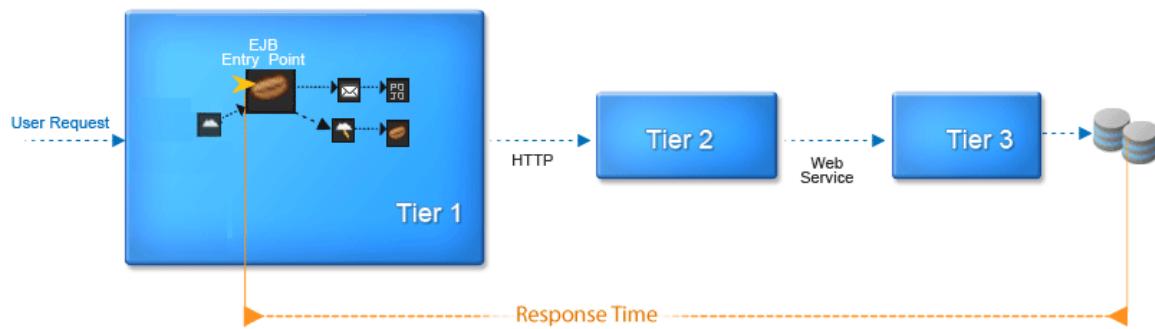
## EJB Entry Points

- EJB-Based Business Transactions
- Default Naming for EJB Entry Points
  - To Enable the Auto-discovery for EJB Transactions
- Custom Match Rules for EJB based Transactions
- Exclude Rules for EJB Transactions

This topic describes how to configure transaction entry points for the EJB based requests.

### EJB-Based Business Transactions

AppDynamics allows you to configure an EJB-based transaction entry point on either the bean name or method name. The response time for the EJB transaction is measured when the EJB entry point is invoked.



### Default Naming for EJB Entry Points

AppDynamics automatically names all the EJB transactions <EJBNamespace>.<MethodName>. By default, automatic transaction discovery for EJB transactions is turned off. To get visibility into these transactions, enable the auto-discovery for EJB based transactions explicitly.

Keep in mind the following before you enable auto-discovery for EJB based transactions:

- If the EJBs use Spring Beans on the front-end, the transaction is discovered at the Spring layer and the response time is measured from the Spring Bean entry point. This is because AppDynamics supports distributed transaction correlation.
- AppDynamics groups all the participating EJB-based transactions (with remote calls) in the same business transaction. However, if your EJBs are invoked from a remote client where the App Server Agent is not deployed, these EJBs are discovered as new business transactions.

### To Enable the Auto-discovery for EJB Transactions

1. Access the transaction detection configurations screen and select the tier to configure. See [To Access Business Transaction Detection Configuration](#).

2. In the EJB entry in the Entry Points section check the Automatic Transaction Detection check box.

Java - Transaction Detection .NET - Transaction Detection

▼ Entry Points

Type	Transaction Monitoring	Automatic Transaction Detection
Servlet	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically <input type="checkbox"/> Enable Servlet Filter Detection
Struts Action	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically Transactions will be named: ActionName
Web Service	<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Discover Transactions automatically Transactions will be named: ServiceName
POJO	<input checked="" type="checkbox"/> Enabled	Any Java method can be the entry point for a transaction. The class to which the method belongs will be used to pick up transactions. The class can be picked using different parameters like name, the interfaces it implements, or its annotations.
Spring Bean	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Discover Transactions automatically Transactions will be named: BeanName
EJB	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Discover Transactions automatically Transactions will be named: EJBNamespace

## Custom Match Rules for EJB based Transactions

If you are not getting the required visibility with auto-discovered transactions, create a custom match rule for a EJB based transaction. See [Custom Match Rules](#).

The following example creates a custom match rule for the receiveOrder method in the TrackOrder bean. The transactions are named "ACME\_EJB.TrackOrder.receiveOrder".

New Business Transaction Match Rule - EJB

Name	ACME_EJB	<input type="button" value="?"/>		
Enabled	<input checked="" type="checkbox"/>			
Business Transaction Match Criteria				
<input checked="" type="checkbox"/>	EJB Name	Equals	TrackOrder	<input type="button" value=""/>
<input checked="" type="checkbox"/>	Method	Equals	receiveOrder	<input type="button" value=""/>
<input type="checkbox"/>	EJB Type	Message Driven		<input type="button" value=""/>
<input type="checkbox"/>	Class Name	Equals		<input type="button" value=""/>
<input type="checkbox"/>	Extends	Equals		<input type="button" value=""/>
<input type="checkbox"/>	Implements	Equals		<input type="button" value=""/>
<input type="button" value="Cancel"/> <input type="button" value="Create Custom Match Rule"/>				

In addition to the bean and method names, other match criteria that could be used to define the transaction are the EJB type, class

name, superclass name and interface name.

## Exclude Rules for EJB Transactions

To exclude specific EJB transactions from detection add an exclude rule. See [Exclude Rules](#). The criteria for EJB exclude rules are the same as those for custom match rules.

## POCO Entry Points

- Default Discovery of POCO Transactions
- Defining a POCO Entry Point
  - Example POCO Entry Point
    - To specify a POCO entry point
  - POCO Transaction as a Background Task
- Further Refining POCO-Based Transactions using "Splitting"
  - To configure transaction splitting
  - To configure exclude rules

Some business processing is not implemented using common or popular frameworks. Your application may perform processing in all types of containers. It may be using a framework that AppDynamics does not automatically detect. You can specify entry points using custom match rules for POCOs (Plain Old C++ Objects). Once defined, AppDynamics measures performance data for POCO transactions as for any other transactions.

Define the custom match rule on the class/method that is the most appropriate entry point for the transaction. Someone who is familiar with your application code should help make this determination.

## Default Discovery of POCO Transactions

By default, the AppDynamics discovery mechanism for POCO-based requests captures ... and displays them in the [Business Transactions List](#). AppDynamics identifies the POCO-based transactions using the class name and method name convention:

```
<Class Name>.<Method Name>.
```

When AppDynamics does not automatically discover a POCO transactions, it may be because....?... the POCO does not implement a predefined interface or because it does not have a predefined annotation...?

AppDynamics' default discovery rules for.... takes precedence over the POCO-based transaction discovery...?

## Defining a POCO Entry Point

The POCO entry point is the method that starts the transaction. Choose a method that begins and ends every time the specific business transaction is invoked. You use custom match rules to enable AppDynamics to discover the transaction.

You can refine the naming of a POJO-based transaction by transaction splitting.

## Example POCO Entry Point

For example, consider the following method execution sequence: REDO CODE IF APPLICABLE

```
REDO CODE IF APPLICABLE

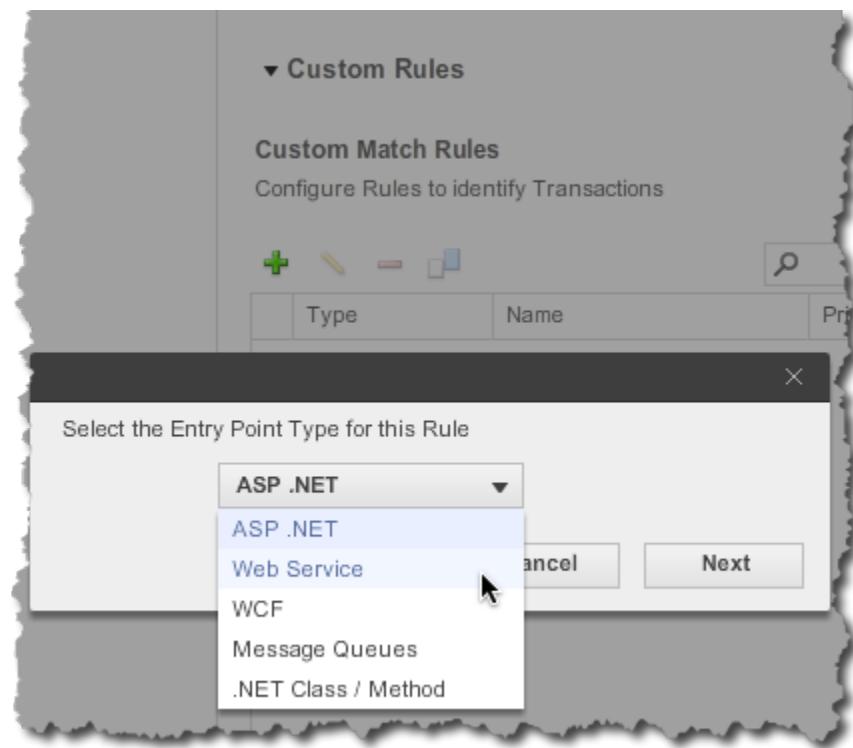
com.foo.threadpool.WorkerThread.run()
    calls com.foo.threadpool.WorkerThread.runInternal()
        calls com.foo.Job.run()
```

The first two calls to run() method are the blocking methods that accept a job and invoke it.

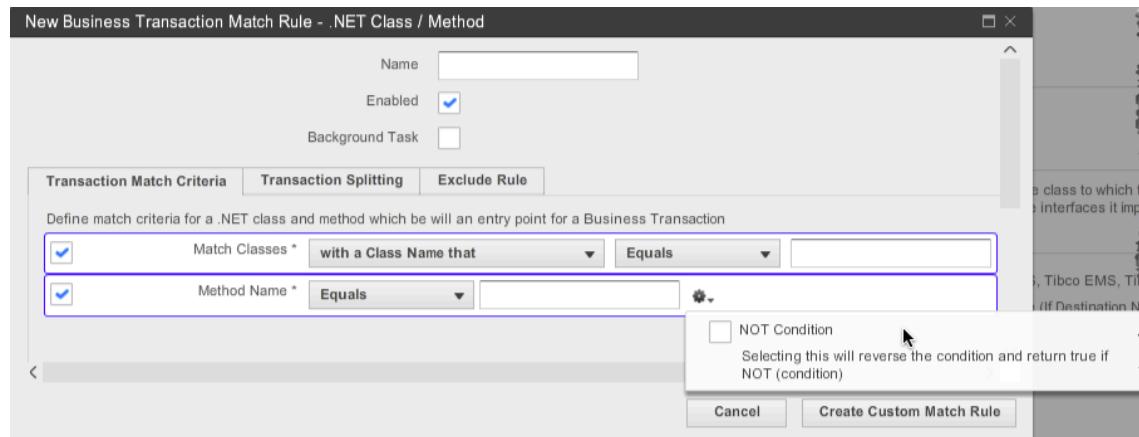
The Job.run() method is the actual unit of work, because Job is executed every time the business transaction is invoked and finishes when the business transaction finishes.

### To specify a POCO entry point

1. Click **Configure -> Instrumentation -> Transaction Detection**.
2. Click the **.NET - Transaction Detection** tab.
2. From the application and tier list at the left, select either:
  - an application, to configure at the level of the entire business application.
  - a tier, to configure at the tier level. At the tier level click **Use Custom Configuration for this Tier**. AppDynamics copies the application configuration to the tier level so that you can modify it for the tier.
3. In the Custom Rules panel under Custom Match Rules, click **Add** (the plus sign).



4. From the dropdown menu, select **\*.NET Class/Method**.



5. Name the match rule.

- By default the rule is Enabled; you can disable it later if needed
- By default it is not a background task; see [POCO Transaction as a Background Task](#).

6. In the Transaction Match Criteria, use the dropdowns to specify the class name:

- The type of class
- The match condition
- A string

7. If also needed, use the dropdowns to specify the method name:

- The match condition
- A string

X. If configuring an application level, click **Configure all Tiers to use this Configuration**.

## **POCO Transaction as a Background Task**

You can indicate that a POCO transaction runs as a background task by checking the Background Task check box in the Match Rule window.

When a request runs as a background task, AppDynamics reports only Business Transaction metrics for the request. It does not aggregate response time and calls metrics at the tier and application levels for background tasks. This ensures that background tasks do not distort the baselines for the business application. Also, you can set a separate set of thresholds for background tasks.

## **Further Refining POCO-Based Transactions using "Splitting"**

When you create a custom rule for POCO transactions all of the qualifying transactions follow the custom rule. In some situations you may need to further refine the discovery rules. For example:

- When you need to identify a transaction based on the value of a parameter in the entry point method.
- When multiple classes have the same method and are instrumented using the same rule, when the method is invoked the class name of the instance being invoked can be used to classify the request.

This process is called "transaction splitting".

In addition, if there are some transactions that result from the custom match and splitting rules that you want to ignore, you can specify custom exclude rules.

### **To configure transaction splitting**

New Business Transaction Match Rule - .NET Class / Method

Enabled

Background Task

**Transaction Match Criteria**    **Transaction Splitting**    **Exclude Rule**

Split .NET class / Method Transactions using one of the following mechanisms...

Use a method parameter to name Transactions  
Parameter Index

Property or Field chain to run on method parameter  getX().getY()

Use the .NET class / Method Object Instance to name Transactions  
Getter Chain to run on .NET class / Method Object Instance  getX().getY()

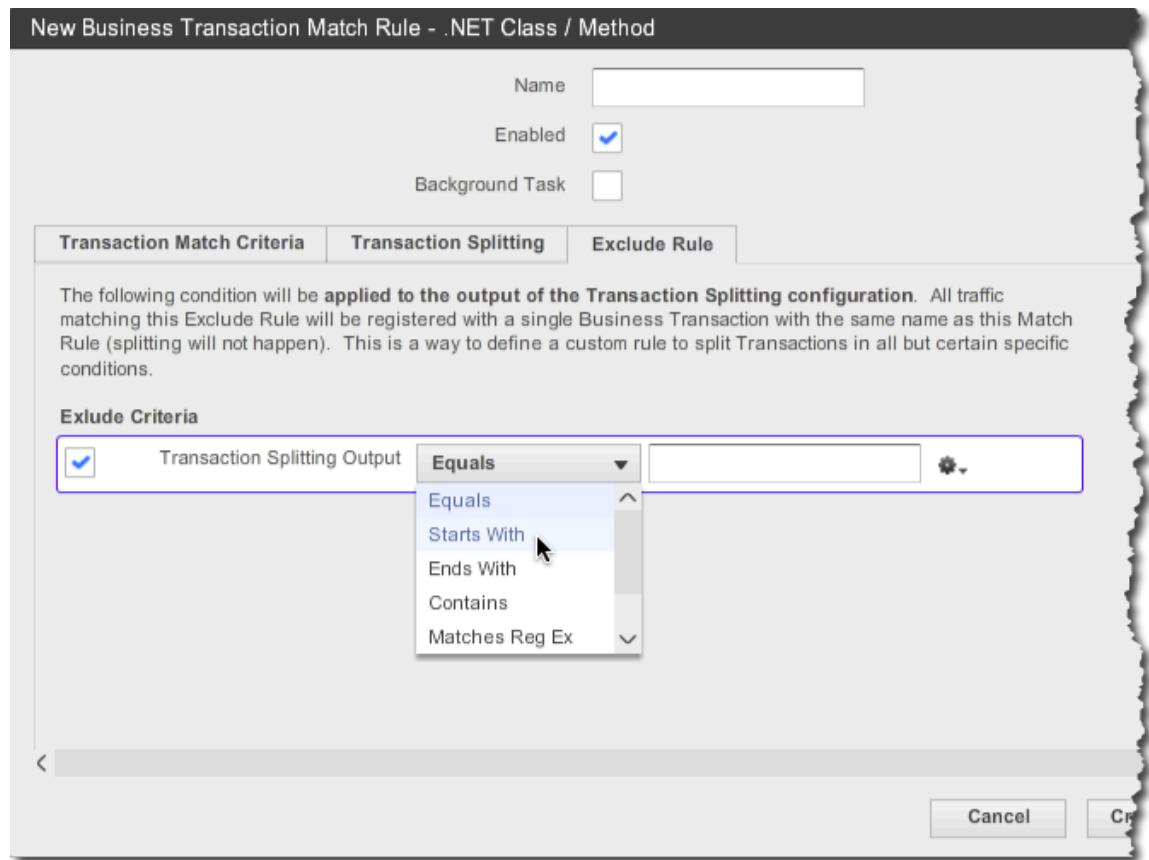
Use the Full Class Name in Transaction names

Use the Simple Class Name in Transaction Names

Use the Thread ID in Transaction names

Use the Thread Name in Transaction names

To configure exclude rules



## Getter Chains in Java Configurations

- Separators in Getter Chains
- Getter Chain Examples
- Curly Braces Enclosing Getter Chains
- Learn More

This topic provides some guidance and examples of the correct syntax for getter chains in AppDynamics configurations.

### Separators in Getter Chains

The following special characters are used as separators:

- comma (,) for separating parameters
- forward slash (/) for separating a type declaration from a value in a parameter
- Dot (.) for separating the methods and properties in the getter chain

If a slash or a comma character is used in a string parameter, use the backslash ()escape character.

If a literal dot is used in a string parameter, use the backslash escape character before the dot.

### Getter Chain Examples

- Getter chain with integer parameters in the substring method using the forward slash as the type separator:

```
getAddress(appdynamics, sf).substring(int/0, int/10)
```

- Getter chain with various non-string parameter types:

```
getAddress(appdynamics, sf).myMethod(float/0.2, boolean/true, boolean/false, int/5)
```

- Getter chain with forward slash escaped; escape character needed here for the string parameter:

```
getUrl().split(\/) # node slash is escaped by a backward slash
```

- Getter chain with an array element:

```
getUrl().split(\/).[4]
```

- Getter chain with multiple array elements separated by commas:

```
getUrl().split(\/).[1,3]
```

- Getter chain using backslash to escape the dot in the string parameter; the call is getParam (a.b.c).

```
getAddress.getParam(a\.\.b\.\.c\.)
```

- In the following getter chain, the first dot requires an escape character because it is in a string method parameter (inside the parentheses). The second dot does not require an escape character because it is not in a method parameter (it is outside the parentheses).

```
getName(suze\.\smith).getClass().getSimpleName()
```

- The following getter chain is from a transaction splitting rule on URIs that use a semicolon as a delimiter; for example: /my-webapp/xyz;jsessionid=BE7F31CC0235C796BF8C6DF3766A1D00?act=Add&uid=c42ab7ad-48a7-4353-bb11-0df  
The getter chain splits on the API name, so the resulting split transactions are "API.abc", API."xyz" and so on.

The call gets the URL using getRequestURI() and then splits it using the escaped forward slash. From the resulting array it takes the third entry (as the split treats the first slash as a separator) and inserts what before the slash (in this case, nothing) into the first entry. Then it splits this result using the semicolon, getting the first entry of the resulting array, which in this case contains the API name.

```
getRequestURI().split(\/).[2].split(;).[0]
```

## Curly Braces Enclosing Getter Chains

In most cases curly braces are not used to enclose getter chains in AppDynamics configurations. An exception is the use of a getter chain in a custom expression on the HttpServletRequest object.

Custom expressions on the HTTP request are configurable in the Java Servlet Transaction Naming Configuration window and in the Split Transactions Using Request Data tab of the servlet custom match and exclude rules. In these cases, curly braces are required to delineate the boundaries of the getter chains.

Business Transaction Match Rule - Servlet

Name	<input type="text" value="API"/>
Enabled	<input type="checkbox"/>
Priority	<input type="text" value="10"/> <a href="#">?</a>
<input checked="" type="radio"/> Transaction Match Criteria <input type="radio"/> Split Transactions Using Request Data <input type="radio"/> Split Transactions Using Payload	
<input type="radio"/> Use the first <input type="text" value="1"/> segments in Transaction names <input type="radio"/> Use the last <input type="text" value="1"/> segments in Transaction names <input type="radio"/> Use URI segment(s) in Transaction names Segment Numbers <input type="text" value="1,2,3,4"/> <i>Enter a comma separated list of parameter numbers (e.g. 1,3,4)</i> <input type="radio"/> Use a parameter value in Transaction names Parameter Name <input type="text"/> <input type="radio"/> Use a header value in Transaction names Header Name <input type="text"/> <input type="radio"/> Use a cookie value in Transaction names Cookie Name <input type="text"/> <input type="radio"/> Use a session attribute value in Transaction names Session Attribute Key <input type="text"/> <input type="radio"/> Use the request method (GET/POST/PUT) in Transaction names <input type="radio"/> Use the request host in names <input type="radio"/> Use the request originating address in Transaction names <input checked="" type="radio"/> Apply a custom expression on HttpServletRequest and use the result in Transaction Names <a href="#">Explain This</a> <code> \${getRequestURI().split('/')[2].split('.')[0]} </code>	
<a href="#">Cancel</a> <a href="#">Save</a>	

**Curly braces required here.**

Getter chains in custom expressions on the HTTP request in diagnostic data collector should also be enclosed in curly braces:

Transaction Detection    Backend Detection    Error Detection    **Diagnostic Data Collectors**    Call Graph Settings    >>

Create HTTP Request Gatherer

Specify the names of the parameter/cookie values to be collected. The value will be displayed in the Transaction Snapshot against the display name chosen here.

**Method Invocation**

Any method invocation or part of a method might tell you something about the context of a transaction.

Name   Apply to new Business Transactions

**HTTP Parameters**

Display Name	HTTP Parameter Name

**HTTP Request Attributes**

URL  
 Session ID  
 User Principal  
 Get User Principal by `httpServletRequest.getUserPrincipal().toString()`.  
 Get User Principal by evaluating a custom expression on the `HttpServletRequest`:  
 `${getHeader(user-agent)}`   
*Enter a custom expression to be applied on the `HttpServletRequest` object. [?](#)*

**Curly braces required here.**

## Learn More

- Configure Business Transaction Detection
- Configure Data Collectors

# Java Server-Specific Installation Settings

**!** If you are a Self-Service Trial user, add the App Agent for Java `javaagent` argument to your JVM start script where `<my-app-jvm1>` is the name you use for the application running on that JVM.

```
-javaagent:<agent_home>/javaagent.jar=uniqueID=<my-app-jvm1>
```

See [Name Business Applications, Tiers, and Nodes](#).

## Apache Cassandra Startup Settings

- To add the javaagent command in a Windows environment
- To add the javaagent command in a Linux environment

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to the cassandra (Linux) or cassandra.bat (Windows) file.

### To add the javaagent command in a Windows environment

1. Open the apache-cassandra-x.x.x\bin\cassandra.bat file.
2. Add the AppDynamics javaagent to the JAVA\_OPTS variable. Make sure to include the drive in the full path to the App Server agent directory.

```
-javaagent:<agent_home>\javaagent.jar
```

For example:

```
set JAVA_OPTS=-ea  
-javaagent:C:\appdynamics\agent\javaagent.jar  
-javaagent:"%CASSANDRA_HOME%\lib\jamm-0.2.5.jar  
. . .  
. . .
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

```
-javaagent:<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>
```

3. Restart the Cassandra server. The Cassandra server must be restarted for the changes to take effect.

### To add the javaagent command in a Linux environment

1. Open the apache-cassandra-x.x.x/bin/cassandra.in.sh file.
2. Add the javaagent argument at the top of the file:

```
JVM_OPTS=-javaagent:<agent_home>/javaagent.jar
```

For example:

```
JVM_OPTS=-javaagent:/home/software/appdynamics/agent/javaagent.jar
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

3. Restart the Cassandra server for the changes to take effect.

## Apache Tomcat Startup Settings

- To add the javaagent command in a Windows environment
- To add the javaagent command in a Linux environment

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to your Tomcat catalina.sh or catalina.bat file.

If you are using Tomcat as a Windows service, see [Tomcat as a Windows Service Configuration](#).

### To add the javaagent command in a Windows environment

1. Open the **catalina.bat** file, located at <apache\_version\_tomcat\_install\_dir>\bin.
2. Add following javaagent argument to the beginning of your application server start script.

```
if "%1"=="stop" goto skip_agent
set JAVA_OPTS=%JAVA_OPTS% -javaagent:"Drive:<agent_home>\javaagent.jar"
:skip_agent
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

```
-javaagent:"Drive:<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>"
```

The javaagent argument references the full path to the App Server Agent installation directory, including the drive. For details see the screen captures.

2a. Sample Tomcat 5.x catalina.bat file

```
catalina.bat
88 rem Add an extra jar file to CLASSPATH
89 rem Note that there are no quotes as we do not want to introduce random
90 rem quotes into the CLASSPATH
91 if "%CLASSPATH%" == "" goto emptyClasspath
92 set CLASSPATH=%CLASSPATH%
93 :emptyClasspath
94 set CLASSPATH=%CLASSPATH%\%CATALINA_HOME%\bin\bootstrap.jar
95
96 if not "%CATALINA_BASE%" == "" goto getBase
97 set CATALINA_BASE=%CATALINA_HOME%
98 :getBase
99
100 if not "%CATALINA_TMPDIR%" == "" goto getTmpdir
101 set CATALINA_TMPDIR=%CATALINA_BASE%\temp
102 :getTmpdir
103
104 if not exist "%CATALINA_HOME%\bin\tomcat-juli.jar" goto noJuli
105 set JAVA_OPTS=%JAVA_OPTS% -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djava.util.logging.config.file=%CATALINA_BASE%\tomcat-juli
106 :noJuli
107
108 if "%1"=="stop" goto skip_agent
109 set JAVA_OPTS=%JAVA_OPTS% -javaagent:"D:\Appdynamics\192\javaagent.jar"
110 :skip_agent
111
112 rem ----- Execute The Requested Command -----
113
114 echo Using CATALINA_BASE: %CATALINA_BASE%
115 echo Using CATALINA_HOME: %CATALINA_HOME%
116 echo Using CATALINA_TMPDIR: %CATALINA_TMPDIR%
117 if "%1" == "debug" goto use_jdk
118 echo Using JRE_HOME: %JRE_HOME%
119 goto java_dir_displayed
120 :use_jdk
121 echo Using JAVA_HOME: %JAVA_HOME%
122 :java_dir_displayed
123 echo Using CLASSPATH: %CLASSPATH%
124
125 set _EXECJAVA=%_RUNJAVA%
126 set MAINCLASS=org.apache.catalina.startup.Bootstrap
127 set ACTION=start
128 set SECURITY_POLICY_FILE=
129 set DEBUG_OPTS=
130 set JPDA=
131
132 if not "%1" == "jpda" goto noJPDA
133 set JPDA=jpda
```

2b. Sample Tomcat 6.x catalina.bat file

```

118
119 if not "%CATALINA_BASE%" == "" goto gotBase
120 set CATALINA_BASE=%CATALINA_HOME%
121 :gotBase
122
123 if not "%CATALINA_TMPDIR%" == "" goto gotTmpdir
124 set CATALINA_TMPDIR=%CATALINA_BASE%\temp
125 :gotTmpdir
126
127 if not "%LOGGING_CONFIG%" == "" goto noJuliConfig
128 set LOGGING_CONFIG=DnLog
129 if not exist "%CATALINA_BASE%\conf\logging.properties" goto noJuliConfig
130 set LOGGING_CONFIG=Djava.util.logging.config.file="%CATALINA_BASE%\conf\logging.properties"
131 :noJuliConfig
132 set JAVA_OPTS=%JAVA_OPTS% %LOGGING_CONFIG%
133
134 if not "%LOGGING_MANAGER%" == "" goto noJuliManager
135 set LOGGING_MANAGER=Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
136 :noJuliManager
137 set JAVA_OPTS=%JAVA_OPTS% %LOGGING_MANAGER%
138
139 if "%1"=="stop" goto skip_agent
140     set JAVA_OPTS=%JAVA_OPTS% -javaagent:"D:\Appdynamics\192\javaagent.jar"
141 :skip_agent
142
143
144 rem ----- Execute The Requested Command -----
145
146 echo Using CATALINA_BASE: %CATALINA_BASE%
147 echo Using CATALINA_HOME: %CATALINA_HOME%
148 echo Using CATALINA_TMPDIR: %CATALINA_TMPDIR%
149 if "%1"=="debug" goto use_jdk
150 echo Using JRE_HOME: %JRE_HOME%
151 goto java_dir_displayed
152 :use_jdk
153 echo Using JAVA_HOME: %JAVA_HOME%
154 :java_dir_displayed
155

```

3. Restart the application server. The application server must be restarted for the changes to take effect.

### To add the javaagent command in a Linux environment

1. Open the catalina.sh file located at <apache\_version\_tomcat\_install\_dir>/bin).
2. Add the following commands at the beginning of your application server start script.

```

if [ "$1" = "start" -o "$1" = "run" ]; then
export JAVA_OPTS="$JAVA_OPTS -javaagent:agent_install_dir/javaagent.jar"
fi

```

The javaagent argument references the full path to the App Server Agent installation directory.

**⚠** If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

For details see the screen captures.

2a. Sample Tomcat 5.x catalina.sh file

```

148 have_tty=0
149 if [ "$tty" != "not a tty" ]; then
150     have_tty=1
151 fi
152
153 # For Cygwin, switch paths to Windows format before running java
154 if $cygwin; then
155     JAVA_HOME=`cygpath --absolute --windows "$JAVA_HOME"`
156     JRE_HOME=`cygpath --absolute --windows "$JRE_HOME"`
157     CATALINA_HOME=`cygpath --absolute --windows "$CATALINA_HOME"`
158     CATALINA_BASE=`cygpath --absolute --windows "$CATALINA_BASE"`
159     CATALINA_TMPDIR=`cygpath --absolute --windows "$CATALINA_TMPDIR"`
160     CLASSPATH=`cygpath --path --windows "$CLASSPATH"`
161     JAVA_ENDORSED_DIRS=`cygpath --path --windows "$JAVA_ENDORSED_DIRS"`
162 fi
163
164 # Set juli LogManager if it is present
165 if [ -r "$CATALINA_HOME/bin/tomcat-juli.jar" ]; then
166     JAVA_OPTS="$JAVA_OPTS -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager"
167     LOGGING_CONFIG="-Djava.util.logging.config.file=$CATALINA_BASE/conf/logging.properties"
168 else
169     # Bugzilla 45585
170     LOGGING_CONFIG="-Dnop"
171 fi
172
173 if [ "$1" = "start" -o "$1" = "run" ] ; then
174     export JAVA_OPTS="$JAVA_OPTS -javaagent:/mnt/agenttest/agent192-2/javaagent.jar"
175 fi
176
177 # ----- Execute The Requested Command -----
178
179 # Bugzilla 37848: only output this if we have a TTY
180 if [ $have_tty -eq 1 ]; then
181     echo "Using CATALINA_BASE: $CATALINA_BASE"
182     echo "Using CATALINA_HOME: $CATALINA_HOME"
183     echo "Using CATALINA_TMPDIR: $CATALINA_TMPDIR"
184     if [ "$1" = "debug" ] ; then
185         echo "Using JAVA_HOME: $JAVA_HOME"
186     else
187         echo "Using JRE_HOME: $JRE_HOME"
188     fi
189 fi

```

## 2b. Sample Tomcat 6.x catalina.sh file

```

199 fi
200
201 # Set juli LogManager config file if it is present and an override has not been issued
202 if [ -z "$LOGGING_CONFIG" ]; then
203     if [ -r "$CATALINA_BASE/conf/logging.properties" ]; then
204         LOGGING_CONFIG="-Djava.util.logging.config.file=$CATALINA_BASE/conf/logging.properties"
205     else
206         # Bugzilla 45585
207         LOGGING_CONFIG="-Dnop"
208     fi
209 fi
210
211 if [ -z "$LOGGING_MANAGER" ]; then
212     JAVA_OPTS="$JAVA_OPTS -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager"
213 else
214     JAVA_OPTS="$JAVA_OPTS $LOGGING_MANAGER"
215 fi
216
217 if [ "$1" = "start" -o "$1" = "run" ] ; then
218     export JAVA_OPTS="$JAVA_OPTS -javaagent:/mnt/agenttest/agent192/javaagent.jar"
219 fi
220
221 # ----- Execute The Requested Command -----
222
223 # Bugzilla 37848: only output this if we have a TTY
224 if [ $have_tty -eq 1 ]; then
225     echo "Using CATALINA_BASE: $CATALINA_BASE"
226     echo "Using CATALINA_HOME: $CATALINA_HOME"
227     echo "Using CATALINA_TMPDIR: $CATALINA_TMPDIR"
228     if [ "$1" = "debug" ] ; then
229         echo "Using JAVA_HOME: $JAVA_HOME"
230     else
231         echo "Using JRE_HOME: $JRE_HOME"
232     fi
233     echo "Using CLASSPATH: $CLASSPATH"
234 fi
235
236 if [ "$1" = "jpdas" ] ; then
237     if [ -z "$JPDA_TRANSPORT" ]; then
238         JPDA_TRANSPORT="socket,bind=1099,server=y"
239     fi
240 fi

```

## 3. Restart the application server. The application server must be restarted for the changes to take effect.

## Tomcat as a Windows Service Configuration

- To install the javaagent as a Tomcat Windows service

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to your Tomcat properties.

If you are not running Tomcat as a Windows service, see [Apache Tomcat Startup Settings](#).

### To install the javaagent as a Tomcat Windows service

These instructions apply to Apache Tomcat 6.x or later versions.

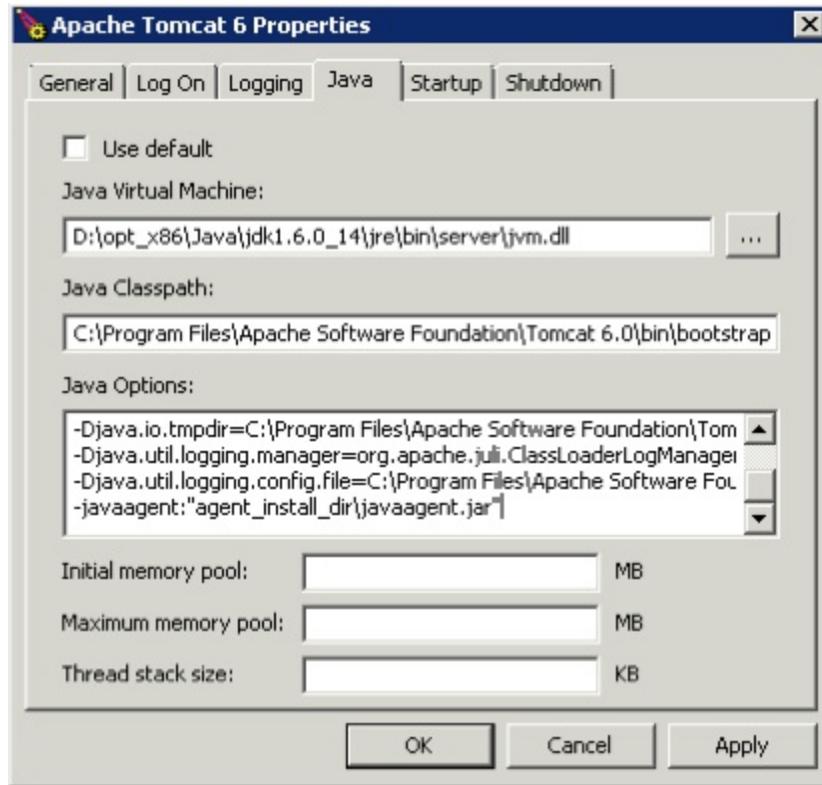
1. Ensure that you are using administrator privileges.
2. Click **Programs -> Apache Tomcat**.
3. Run **Configure Tomcat**.
4. Click the **Java** tab.
5. In the **Java Options** add:

```
-javaagent:<agent_home>\javaagent.jar
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

```
-javaagent:<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>
```

For details see the following screenshot.



6. Restart the Tomcat service. The application server must be restarted for the changes to take effect.

## Glassfish Startup Settings

- To add the javaagent command in a GlassFish environment
- To verify the Agent configuration
- About AppServer Management Extensions

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option.

### To add the javaagent command in a GlassFish environment

1. If you are using **GlassFish v3.x**, first configure the OSGi containers. For details see [OSGi Infrastructure Configuration](#).
2. Log into the **GlassFish domain** where you want to install the App Server Agent.
3. In the left navigation tree **Common Tasks** section, click **Application Server**. The Application Server Settings dialog opens.
4. In the **JVM Settings** tab, click **JVM Options**.
5. Click **Add JVM Option** and add an entry for the javaagent argument. The javaagent argument contains the full path, including the drive, of the App Server Agent installation directory.

```
-javaagent:<drive>:\<agent_home>\javaagent.jar
```

**!** If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm> is the name you use for the application running on that JVM.

```
-javaagent:<drive>:\<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>
```

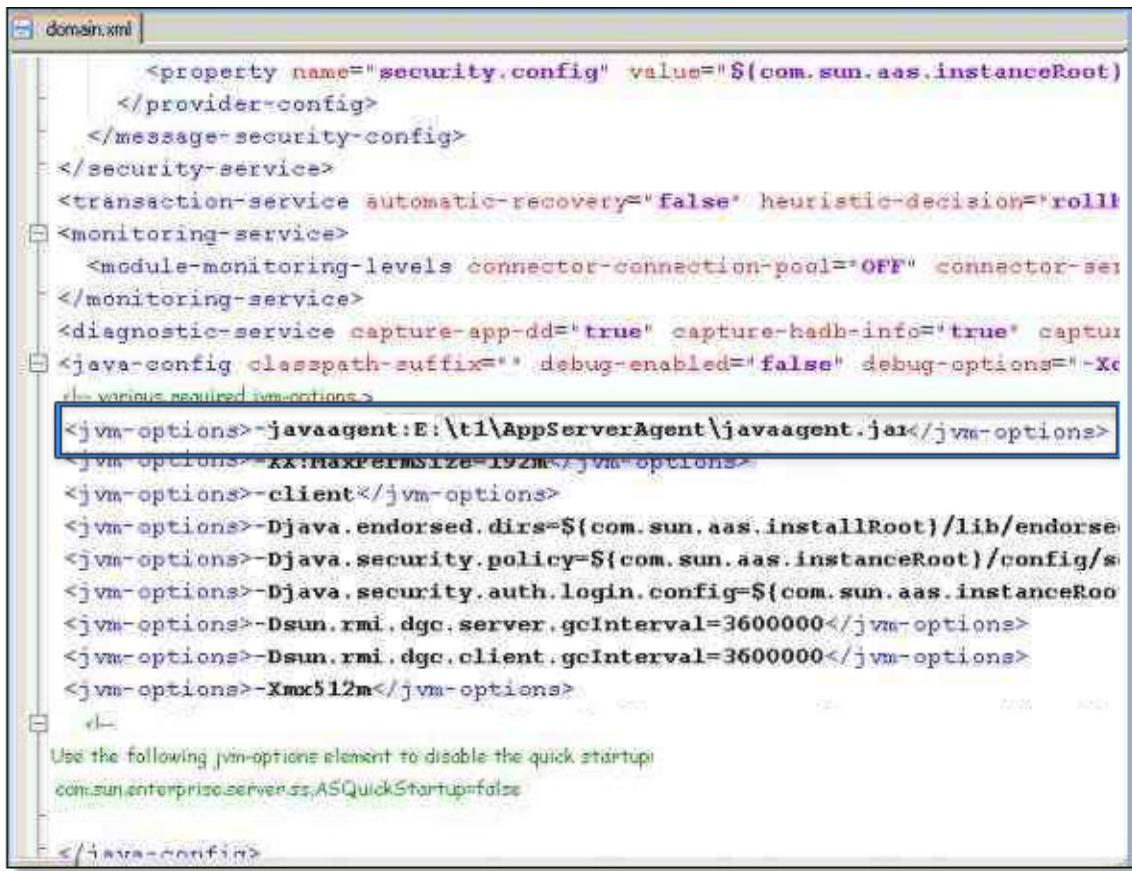
The screenshot shows the Sun GlassFish Enterprise Server Administration Console. At the top, the URL bar displays "User: admin Domain: domain1 Server: localhost". Below the URL bar, the title "Sun GlassFish™ Enterprise Server" is visible. A green oval highlights the "Domain: domain1" part of the URL. In the left sidebar, under "Common Tasks", the "Application Server" node is highlighted with a green oval. A blue arrow points down from this node to a detailed view of the "JVM Options" tab. The "JVM Options" tab is selected, indicated by a blue border. Another blue arrow points down to the "Options (18)" table. The table has columns for "Value" and checkboxes. One row in the table is highlighted with a blue box, showing the value "-javaagent:E:\t1\AppServerAgent\javaagent.jar". To the right of the table is a "Save" button.

Value
<input type="checkbox"/> -javaagent:E:\t1\AppServerAgent\javaagent.jar
<input type="checkbox"/> -XX:MaxPermSize=192m
<input type="checkbox"/> -client

6. Restart the application server. The application server must be restarted for the changes to take effect.

### To verify the Agent configuration

To verify this configuration, look at the domain.xml file located at <glassfish\_install\_dir>\domains\<domain\_name>. The domain.xml file should have an entry as shown in the following screenshot.



```

<domain.xml>
    <provider-config>
        </provider-config>
    </message-security-config>
</security-service>
<transaction-service automatic-recovery="false" heuristic-decision="rollb
<monitoring-service>
    <module-monitoring-levels connector-connection-pool="OFF" connector-se
</monitoring-service>
<diagnostic-service capture-app-dd="true" capture-hadb-info="true" captur
<java-config classpath-suffix="" debug-enabled="false" debug-options="-Xc
    <various required configurations>
<jvm-options>-javaagent:E:\t1\AppServerAgent\javaagent.jar</jvm-options>
    <jvm-options>-XX:MaxPermSize=192M</jvm-options>
    <jvm-options>-client</jvm-options>
    <jvm-options>-Djava.endorsed.dirs=${com.sun.aas.instanceRoot}/lib/endorse
    <jvm-options>-Djava.security.policy=${com.sun.aas.instanceRoot}/config/s
    <jvm-options>-Djava.security.auth.login.config=${com.sun.aas.instanceRoo
    <jvm-options>-Dsun.rmi.dgc.server.gcInterval=3600000</jvm-options>
    <jvm-options>-Dsun.rmi.dgc.client.gcInterval=3600000</jvm-options>
    <jvm-options>-Xmx512m</jvm-options>
</various required configurations>
</java-config>
</domain.xml>

```

Use the following jvm-options element to disable the quick startup  
com.sun.enterprise.server.ss.ASQuickStartup=false

## About AppServer Management Extensions

AppDynamics does not support Glassfish AMX. AMX MBeans do not appear in the MBean Browser.

## IBM WebSphere Startup Settings

- To add the javaagent command in a WebSphere environment
- To verify the Agent configuration
- Security permissions requirement

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option.

### To add the javaagent command in a WebSphere environment

1. Log in to the **Administrator** console of the WebSphere node where you want to install the App Server Agent.
2. In the left navigation tree, click **Servers -> Application servers**.
3. Click the name of your server in the list of servers.

For quick access, place your bookmarks here in the bookmarks bar.

Integrated Solutions Console Welcome jpowar

Help | Logout IBM

View: All tasks

- Welcome
- Guided Activities**
- Servers
  - Application servers
  - Web servers
  - WebSphere MQ servers
- Applications
  - Enterprise Applications
  - Install New Application
- Resources
- Security
- Environment

Application servers

**Application servers**

Use this page to view a list of the application servers in your environment. You can also use this page to change the status of a specific server.

**Preferences**

Name	Node	Version
server1	ww-84f6cf8ea82aNode01	Base 6.1.0.0

Total 1



4. In the "Configuration" tab, click **Java and Process Management**.

**Application servers > server1**

Use this page to configure an application server. An application server is a server that provides services required to run enterprise applications.

**Runtime Configuration**

**General Properties**

Name: server1

Node name: rajendraNode01

Run in development mode

Parallel start

Start components as needed

Access to internal server classes: Allow

**Server-specific Application Settings**

Classloader policy: Multiple

Class loading mode: Classes loaded with parent class loader first

**Container Settings**

- [Session management](#)
- + [SIP Container Settings](#)
- + [Web Container Settings](#)
- + [Portlet Container Settings](#)
- + [EJB Container Settings](#)
- + [Container Services](#)
- + [Business Process Services](#)

**Applications**

- [Installed applications](#)

**Server messaging**

- [Messaging engines](#)
- [Messaging engine inbound transports](#)
- [WebSphere MQ link inbound transports](#)
- [SIB service](#)

**Server Infrastructure**

- + Java and Process Management
- + Administration

**Communications**

- + Ports

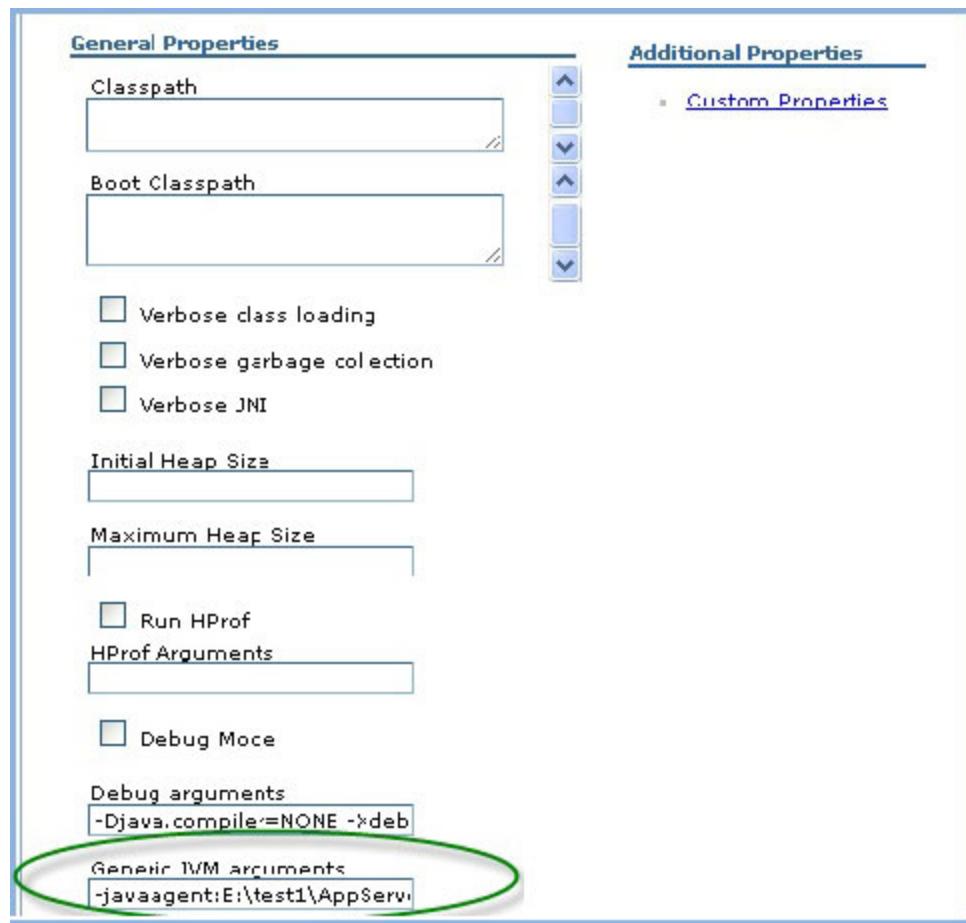
5. Enter the javaagent option with the full path to the AppDynamics javaagent.jar file in the **Generic JVM arguments** field.

-javaagent:<drive>:\<agent\_home>\javaagent.jar

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

-javaagent:<drive>:\<agent\_home>\javaagent.jar=uniqueID=<my-app-jvm1>

6. Click OK.



## To verify the Agent configuration

Verify the configuration settings by checking the server.xml file of the WebSphere node where you installed the App Server Agent. The server.xml file should have this entry:

```
<jvmEntries ...
genericJvmArguments=' -javaagent:E:\test1\AppServerAgent\javaagent.jar'
cisableJIT="false"/>
```

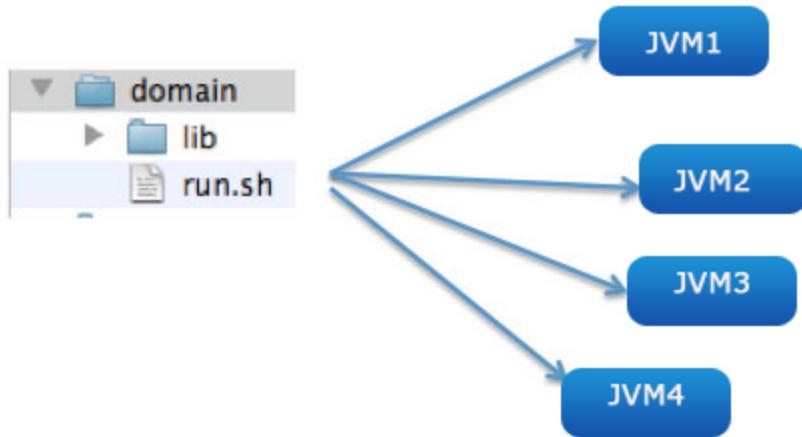
## Security permissions requirement

Full permissions are required for the agent to function correctly with WebSphere. Grant all permissions on both the server level and the profile level.

## App Agent for Java on z-OS or Mainframe Environments Configuration

- To configure the Agent to automatically generate node prefixes
- To remove the nodes that are not alive

In some environments JVMs have transient identity, such as when a single script spawns multiple JVMs.



For example, an environment may consist of WebSphere on IBM Mainframes, using a dynamic workload management feature that spawns new JVMs for an existing application server (called a servant). These JVMs are exact clones of an existing JVM, but each of them has a different process ID. Based on load, any number of additional JVMs may be created.

The App Server Agent can monitor each of these dynamically generated JVMs using the `appdynamics.agent.auto.node.prefix` option. You specify a node name prefix in your JVM startup script.

```
-Dappdynamics.agent.auto.node.prefix=<node name prefix>
```

#### To configure the Agent to automatically generate node prefixes

1. Add the application and tier name to "controller-info.xml" file for each JVM.

**i** These JVMs belong to the **same** tier.

2. Add the `javaagent` argument and system properties (-D options) to the startup script of your JVM processes or to your Java Runtime Environment (JRE):

```
java -javaagent:<Agent-Installation-Directory>/javaagent.jar
-Dappdynamics.agent.reuse.nodeName=true
-Dappdynamics.agent.auto.node.prefix=$nodePrefix
```

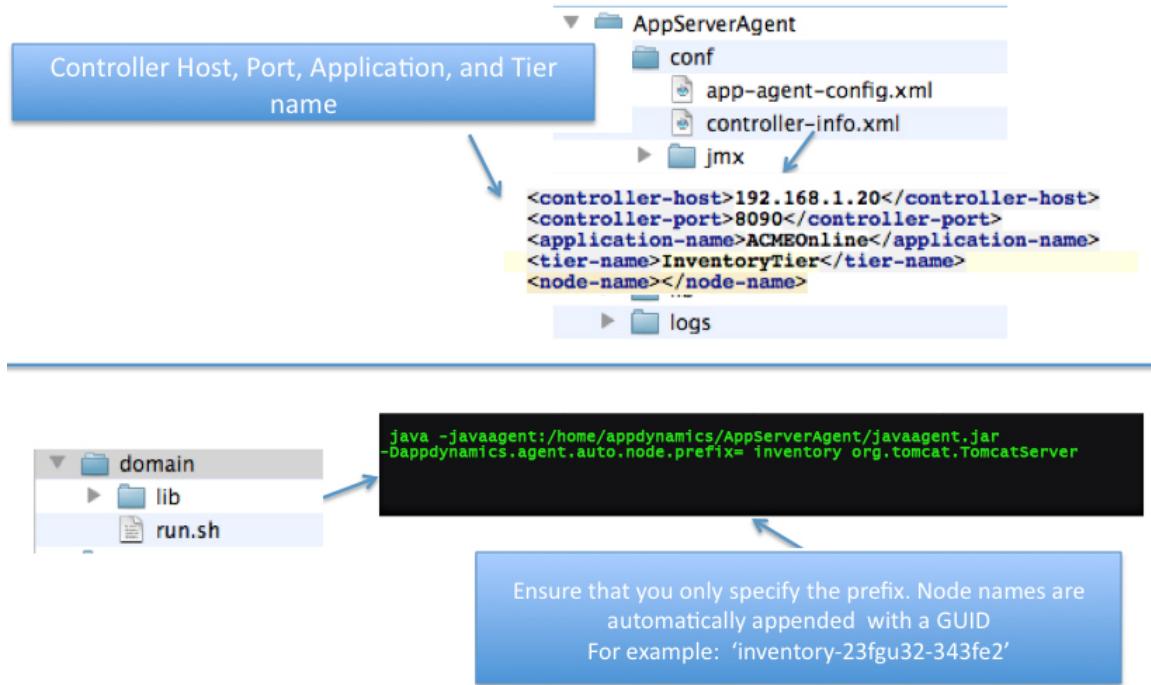
By default, the node names will be based on a serial number maintained by the Controller with a prefix of the tier name. If you need to change the prefix, specify the property `-Dappdynamics.agent.auto.node.prefix=$nodePrefix`.

For example, if no prefix is specified for node on tier ECommerceTier, the node name will be ECommerceTier-1, ECommerceTier-2, etc. If a prefix is provided as in the example, the node names will be customPrefix-1, customPrefix-2, etc.

#### **IMPORTANT:**

- Ensure that you have not specified the node name anywhere (controller-info.xml file or as a system property in your JVMs start-up script).
- Ensure that all these system properties are separated by a white space character.
- These JVMs may or may not be short-lived.

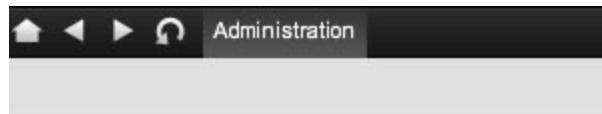
The following illustration shows the sample configuration for ACME Bookstore. This configuration will create unique node names for every instance of the virtual machine starting up in the ACME Bookstore environment.



### To remove the nodes that are not alive

In a typical environment, the nodes are recycled and new nodes get generated. AppDynamics strongly recommends that you remove the dead nodes both from the AppDynamics user interface (UI) as well as from the system.

1. Log in to the Controller administration console.
2. Go to "Controller Settings" section to see the advanced properties for the Controller.



## AppDynamics Administration

### Accounts

Create and Manage Accounts

### Controller Settings

Configure the Controller

3. Set the retention and deletion properties, based on the requirements for your environment. AppDynamics recommends that you set the permanent deletion period at least an hour more than the retention period.

- **node.permanent.deletion.period:** Time (in hours) after which a node that has lost contact with the Controller is deleted permanently from the system.
- **node.retention.period:** Time (in hours) after which a node that has lost contact with the Controller is deleted. In this case, the

AppDynamics UI will not display the node, however the system will continue to retain it.

metrics.write.thread.count	The count of parallel threads to be i	<input type="text" value="1"/>
multitenant.controller	Is the controller running in multi-tier	<input type="text" value="false"/>
node.permanent.deletion.period	Time (in hours) after which a node t	<input type="text" value="720"/>
node.retention.period	Time (in hours) after which a node t	<input type="text" value="500"/>

## JBoss Startup Settings

- To add the javaagent command in a Windows environment
- To add the javaagent command in a Linux environment for JBoss 5.x
- To add the javaagent command in a Linux environment for JBoss AS 6.x
- To add the javaagent command in a Linux environment for JBoss AS 7.x
- To add the javaagent command in a Windows environment for JBoss AS 7.x

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to your JBoss server run.sh or run.bat file.

### To add the javaagent command in a Windows environment

1. Open the server run.bat file, located at <jboss\_version\_install\_directory>\bin.
2. Add the following javaagent argument at the beginning of your app server start script.

```
set JAVA_OPTS=%JAVA_OPTS% -javaagent:<drive>:\<agent_home>\javaagent.jar
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

```
-javaagent:<drive>:\<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>
```

The javaagent argument references the full path of the App Server Agent installation directory, including the drive. For details see the screen captures.

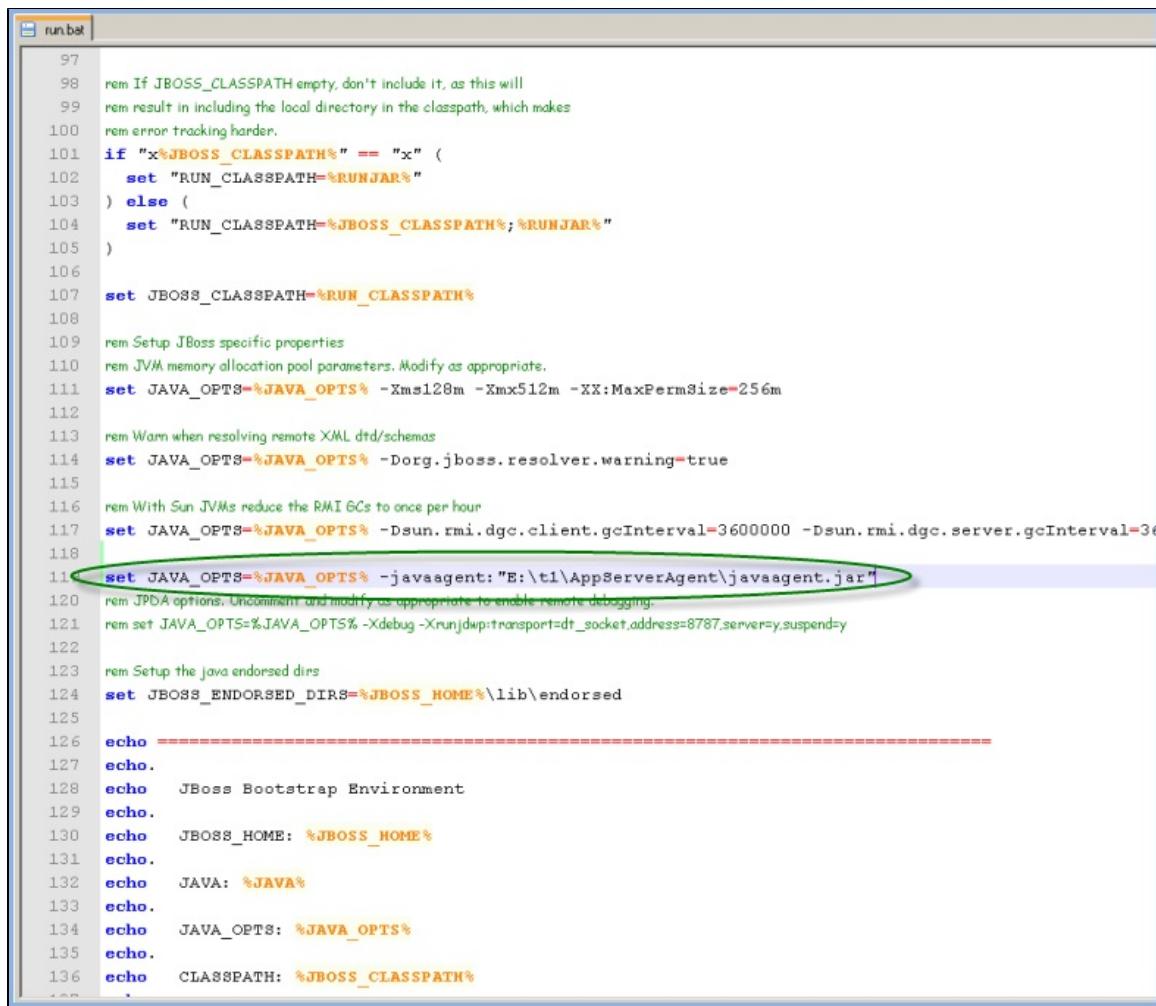
2a. Sample JBoss 4.x run.bat file

```

62
63 rem If JBOSS_CLASSPATH is empty, don't include it, as this will
64 rem result in including the local directory, which makes error tracking
65 rem harder.
66 if "%JBOSS_CLASSPATH%" == "" (
67     set JBOSS_CLASSPATH=%JAVAC_JAR%;%RUNJAR%
68 ) else (
69     set JBOSS_CLASSPATH=%JBOSS_CLASSPATH%;%JAVAC_JAR%;%RUNJAR%
70 )
71
72 rem Setup JBoss specific properties
73 set JAVA_OPTS=%JAVA_OPTS% -Dprogram.name=%PROGNAME%
74 set JBOSS_HOME=%DIRNAME%..
75
76 rem Add -server to the JVM options, if supported
77 "%JAVA%" -version 2>&1 | findstr /I hotspot > nul
78 if not errorlevel == 1 (set JAVA_OPTS=%JAVA_OPTS% -server)
79
80 rem JVM memory allocation pool parameters. Modify as appropriate.
81 set JAVA_OPTS=%JAVA_OPTS% -Xms128m -Xmx512m
82
83 rem With Sun JVMs reduce the RMI GCs to once per hour
84 set JAVA_OPTS=%JAVA_OPTS% -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=360000
85
86 set JAVA_OPTS=%JAVA_OPTS% -javaagent:"E:\t1\AppServerAgent\javaagent.jar"
87 rem JDB options. Uncomment and modify as appropriate to enable remote debugging.
88 rem set JAVA_OPTS=-Xdebug -Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=y %JAVA_OPTS%
89
90 rem Setup the java endorsed dirs
91 set JBOSS_ENDORSED_DIRS=%JBOSS_HOME%\lib\endorsed
92
93 echo -----
94 echo.
95 echo JBoss Bootstrap Environment
96 echo.
97 echo JBOSS_HOME: %JBOSS_HOME%
98 echo.
99 echo JAVA: %JAVA%
echo.
101 echo JAVA_OPTS: %JAVA_OPTS%

```

2b. Sample JBoss 5.x run.bat file



```

97
98 rem If JBOSS_CLASSPATH empty, don't include it, as this will
99 rem result in including the local directory in the classpath, which makes
100 rem error tracking harder.
101 if "%JBoss_CLASSPATH%" == "" (
102     set "RUN_CLASSPATH=%RUNJAR%"
103 ) else (
104     set "RUN_CLASSPATH=%JBoss_CLASSPATH%;%RUNJAR%"
105 )
106
107 set JBoss_CLASSPATH=%RUN_CLASSPATH%
108
109 rem Setup JBoss specific properties
110 rem JVM memory allocation pool parameters. Modify as appropriate.
111 set JAVA_OPTS=%JAVA_OPTS% -Xms128m -Xmx512m -XX:MaxPermSize=256m
112
113 rem Warn when resolving remote XML dtd/schemas
114 set JAVA_OPTS=%JAVA_OPTS% -Dorg.jboss.resolver.warning=true
115
116 rem With Sun JVMs reduce the RMI GCs to once per hour
117 set JAVA_OPTS=%JAVA_OPTS% -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000
118
119 set JAVA_OPTS=%JAVA_OPTS% -javaagent: "E:\t1\AppServerAgent\javaagent.jar"
120 rem JPA options. Uncomment and modify as appropriate to enable remote debugging.
121 rem set JAVA_OPTS=%JAVA_OPTS% -Xdebug -Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=y
122
123 rem Setup the java endorsed dirs
124 set JBoss_ENDORSED_DIRS=%JBoss_HOME%\lib\endorsed
125
126 echo =====
127 echo.
128 echo JBoss Bootstrap Environment
129 echo.
130 echo JBoss_HOME: %JBoss_HOME%
131 echo.
132 echo JAVA: %JAVA%
133 echo.
134 echo JAVA_OPTS: %JAVA_OPTS%
135 echo.
136 echo CLASSPATH: %JBoss_CLASSPATH%

```

3. Restart the application server. The application server must be restarted for the changes to take effect.

### To add the javaagent command in a Linux environment for JBoss 5.x

1. Open the server run.sh file, located at <jboss\_version\_install\_dir>/bin.
2. Add the following javaagent argument to the server start script.

```
export JAVA_OPTS="$JAVA_OPTS -javaagent:<agent_home>/javaagent.jar"
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

2a. Sample JBoss 5.x run.sh file

```

run.sh

130     fi
131
132     # Enable -server if we have Hotspot, unless we can't
133     if [ "x$HAS_HOTSPOT" != "x" ]; then
134         # MacOS does not support -server flag
135         if [ "$darwin" != "true" ]; then
136             JAVA_OPTS="-server $JAVA_OPTS"
137         fi
138         fi
139     fi
140
141     # Setup JBoss specific properties
142     JAVA_OPTS="-Dprogram.name=$PROGNAME $JAVA_OPTS"
143
144     # Setup the java endorsed dirs
145     JBOSS_ENDORSED_DIRS="$JBOSS_HOME/lib/endorsed"
146
147     # For Cygwin, switch paths to Windows format before running java
148     if $cygwin; then
149         JBOSS_HOME=`cygpath --path --windows "$JBOSS_HOME"`
150         JAVA_HOME=`cygpath --path --windows "$JAVA_HOME"`
151         JBOSS_CLASSPATH=`cygpath --path --windows "$JBOSS_CLASSPATH"`
152         JBOSS_ENDORSED_DIRS=`cygpath --path --windows "$JBOSS_ENDORSED_DIRS"`
153     fi
154     export JAVA_OPTS="$JAVA_OPTS -javaagent:/home/AppServerAgent/javaagent.jar"
155     # Display our environment
156     echo ====
157     echo ""
158     echo "  JBoss Bootstrap Environment"
159     echo ""
160     echo "  JBOSS_HOME: $JBOSS_HOME"

```

3. Restart the application server. The application server must be restarted for the changes to take effect.

### To add the javaagent command in a Linux environment for JBoss AS 6.x

1. Open the server run.sh file, located at <jboss\_version\_install\_dir>/bin.
2. Add the following Java environment variables to the server start script.

```

JAVA_OPTS="$JAVA_OPTS
-Djava.util.logging.manager=org.jboss.logmanager.LogManager"
JAVA_ARGS="$JAVA_OPTS
-Dorg.jboss.logging.Logger.pluginClass=org.jboss.logging.logmanager.LoggerPluginIn

```

3. Add the following javaagent argument to the server start script.

```
export JAVA_OPTS="$JAVA_OPTS -javaagent:/agent_install_dir/javaagent.jar"
```

4. Restart the application server. The application server must be restarted for the changes to take effect.

## To add the javaagent command in a Linux environment for JBoss AS 7.x

1. Open the standalone.conf file.
2. Search for the following line in standalone.conf.

```
JBOSS_MODULES_SYSTEM_PKGS="org.jboss.byteman"
```

Add the com.singularity and org.jboss.logmanager packages to that line as follows:

```
JBOSS_MODULES_SYSTEM_PKGS="org.jboss.byteman,com.singularity,org.jboss.logmanager"
```

For JBoss 7.1.1, add the additional packages as shown here:

```
JBOSS_MODULES_SYSTEM_PKGS="org.jboss.byteman,com.appdynamics,com.appdynamics.,com
```

3. Add the following to the end of the standalone.conf file in the JAVA\_OPTION section.

```
-Djava.util.logging.manager=org.jboss.logmanager.LogManager  
-Xbootclasspath/p:<JBoss-DIR>/modules/org/jboss/logmanager/main/jboss-logmanager-
```

Note: The path for the necessary JAR files may differ for different versions. Provide the correct path of these JAR files for your version. If any of the packages are not available with the JBoss ZIP, download the missing package and add it to the path.

4. In the standalone.sh file, add the following javaagent argument.

```
export JAVA_OPTS="$JAVA_OPTS -javaagent:/agent_install_dir/javaagent.jar"
```

above the following section of standalone.sh

```
...  
while true;do  
if [ "$LAUNCH_JBOSS_IN_BACKGROUND" = "X" ]; then  
# Execute the JVM in the foreground  
eval \"$JAVA\" -D\"[Standalone]\"$JAVA_OPTS \  
\"-Dorg.jboss.boot.log.file=$JBOSS_LOG_DIR/boot.log\" \  
\"-Dlogging.configuration=file:$JBOSS_CONFIG_DIR/logging.properties\" \  
-jar \"$JBOSS_HOME/jboss-modules.jar\" \  
...
```

The revised section of your startup script file should look similar to the following image:

```

173 echo ""
174
175 export JAVA_OPTS="$JAVA_OPTS -javaagent:/agent_install_dir/javaagent.jar"
176
177 while true; do
178     if [ "x$LAUNCH_JBOSS_IN_BACKGROUND" = "x" ]; then
179         # Execute the JVM in the foreground
180         eval \'$JAVA\' -D\"[Standalone]\" $JAVA_OPTS \
181             \\"-Dorg.jboss.boot.log.file=$JBOSS_LOG_DIR/boot.log\" \
182             \\"-Dlogging.configuration=file:$JBOSS_CONFIG_DIR/logging.properties\" \
183             \\"-jar \"$JBOSS_HOME/jboss-modules.jar\" \
184             \\"-mp \"$JBOSS_MODULEPATH\" \
185             \\"-jaxpmodule \"javax.xml.jaxp-provider\" \
186             org.jboss.as.standalone \
187             \\"-Djboss.home.dir=\"$JBOSS_HOME\" \
188             \"\$@"
189         JBOSS_STATUS=\$?
190     else

```

5. End User Monitoring (EUM) is not supported on JBoss 7 and you must disable it using the eum-disable-filter-injection agent configuration property. This property is available in AppDynamics App Agent for Java, version 3.5.2 and newer.

- In the Controller UI left navigation pane, select **Servers > App Servers<tier> -> <node>**.
- Click the **Agents** tab and then the **App Server Agent** subtab.
- Click **Configure**.
- In the App Server Agent Configuration, select the node.
- Click Add (the plus symbol) and enter the eum-disable-filter-injection property. Its type is Boolean and its value is "true".
- Click **Save**.

6. Restart the application server. The application server must be restarted for the changes to take effect.

## To add the javaagent command in a Windows environment for JBoss AS 7.x

- Open the bin\standalone.conf file.
- Search for the line JBOSS\_MODULES\_SYSTEM\_PKGS="org.jboss.byteman" and the com.singularity ad org.jboss.logmanager packages to that line as follows:

```
JBOSS_MODULES_SYSTEM_PKGS="org.jboss.byteman,com.singularity,org.jboss.logmanager"
```

- Open the bin\standalone.conf.bat file.

4. Search for the following line:  
[code]  
set "JAVA\_OPTS=%JAVA\_OPTS% -Djboss.modules.system.pkgs=org.jboss.byteman

```
set "JAVA_OPTS=%JAVA_OPTS%
-Djboss.modules.system.pkgs=org.jboss.byteman,com.singularity"
```

- Save the file.
- Open the standalone.bat file.
- Add the following javaagent argument to the standalone.bat file.

```
:RESTART
"%JAVA%" -javaagent:<AGENT-DIR>javaagent.jar %JAVA_OPTS% ^
-Dorg.jboss.boot.log.file=%JBOSS_HOME%\standalone\log\boot.log" ^
-Dlogging.configuration=file:%JBOSS_HOME%\standalone\configuration\logging.properties
^
-jar "%JBOSS_HOME%\jboss-modules.jar" ^
```

8. Save the file.
9. End User Monitoring (EUM) is not supported on JBoss 7 and you must disable it using the eum-disable-filter-injection agent configuration property. This property is available in AppDynamics App Agent for Java, version 3.5.2 and newer.
  - a. In the Controller UI left navigation pane, select **Servers > App Servers<tier> -> <node>**.
  - b. Click the **Agents** tab and then the **App Server Agent** subtab.
  - c. Select the node.
  - d. Click Add (the plus symbol) and enter the eum-disable-filter-injection property. Its type is Boolean and its value is "true".
  - e. Click **Save**.

10. Restart the application server. The application server must be restarted for the changes to take effect.

## Jetty Startup Settings

- To add the javaagent command in a Jetty environment

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to your jetty.sh file.

### To add the javaagent command in a Jetty environment

1. Open the jetty.sh start script file.
2. Add the following javaagent argument to the beginning of the script.

```
java -javaagent:/<agent_home>/javaagent.jar
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

3. Save the script file.
4. Restart the application server for the changes to take effect.

## Oracle WebLogic Startup Settings

- To add the javaagent command in a Windows environment
- To add the javaagent command in a Linux environment

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to your startWebLogic.sh or startWebLogic.cmd file.

### To add the javaagent command in a Windows environment

1. Open the startWebLogic.cmd file, located at <weblogic\_version\_install\_dir>\user\_projects\domains\<domain\_name>\bin.
2. Add following javaagent argument to the beginning of your application server start script.

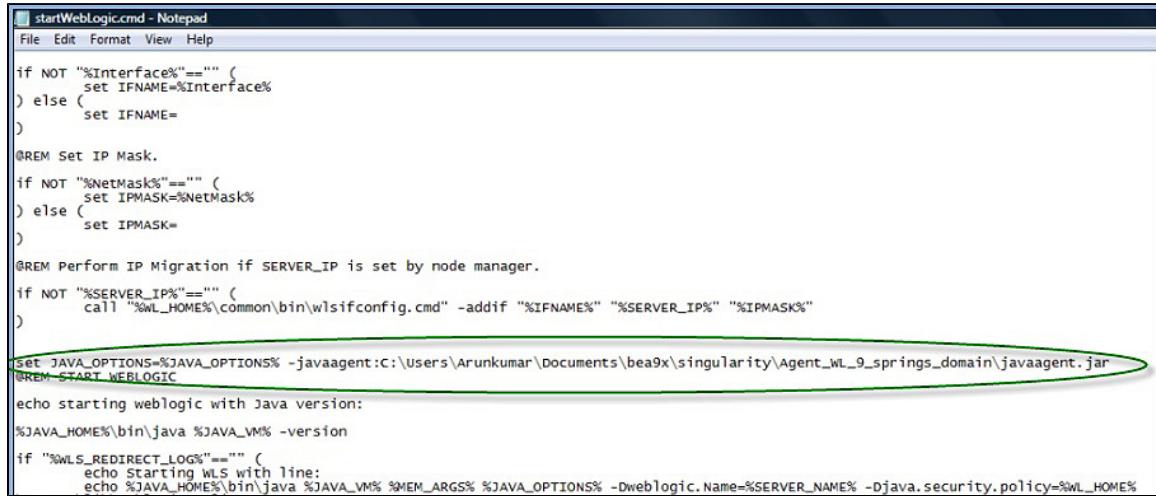
```
set JAVA_OPTIONS=% JAVA_OPTIONS%
-javaagent:<drive>:\<agent_home>\javaagent.jar"
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

```
-javaagent:<drive>:\<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>"
```

The javaagent argument references the full path of the App Server Agent installation directory, including the drive.

## 2a. Sample WebLogic v9.x startWebLogic.cmd file



```
startWebLogic.cmd - Notepad
File Edit Format View Help

if NOT "%Interface%"==""
    set IFNAME=%Interface%
) else (
    set IFNAME=
)

@REM Set IP Mask.
if NOT "%NetMask%"==""
    set IPMASK=%NetMask%
) else (
    set IPMASK=
)

@REM Perform IP Migration if SERVER_IP is set by node manager.
if NOT "%SERVER_IP%"==""
    call "%WL_HOME%\common\bin\wlconfig.cmd" -addif "%IFNAME%" "%SERVER_IP%" "%IPMASK%"

set JAVA_OPTIONS=%JAVA_OPTIONS% -javaagent:C:\users\Arunkumar\Documents\bea9x\singularity\Agent_WL_9_springs_domain\javaagent.jar
@REM START WEBLOGIC

echo starting weblogic with Java version:
%JAVA_HOME%\bin\java %JAVA_VM% -version
if "%WLS_REDIRECT_LOG%"==""
    echo Starting WLS with line:
    echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME% -Djava.security.policy=%WL_HOME%
```

## 2b. Sample WebLogic v10.x startWebLogic.cmd file

```

startWebLogic.cmd
165 ) else (
166     set IPMASK=
167 )
168
169 REM Perform IP Migration if SERVER_IP is set by node manager.
170
171 if NOT "%SERVER_IP%"=="" (
172     call "%WL_HOME%\common\bin\wlsifconfig.cmd" -addif "%IFNAME%" "%SERVER_IP%" "%IPMASK%"
173 )
174
175 set JAVA_OPTIONS=%JAVA_OPTIONS% -javaagent:"E:\t1\AppServerAgent\javaagent.jar"
176 REM START WebLOGIC
177
178 echo starting weblogic with Java version:
179
180 %JAVA_HOME%\bin\java %JAVA_VM% -version
181
182 if "%WLS_REDIRECT_LOG%"=="" (
183     echo Starting WLS with line:
184     echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
185     echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
186 ) else (
187     echo Redirecting output from WLS window to %WLS_REDIRECT_LOG%
188     %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
189 )

```

3. Restart the application server. The application server must be restarted for the changes to take effect.

### To add the javaagent command in a Linux environment

1. Open the startWebLogic.sh file, located at <weblogic\_<version#>\_install\_dir>/user\_projects/domains/<domain\_name>/bin.
2. Add the following lines of code to the beginning of your application server start script.

```
export JAVA_OPTIONS="$JAVA_OPTIONS -javaagent:/agent_home/javaagent.jar"
```

**⚠** If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

The javaagent argument references the full path of the App Server Agent installation directory. For details see the screen captures.

2a. Sample WebLogic v9.x startWebLogic.sh file

```
startWebLogic.sh |  
167 if [ "${SERVER_IP}" != "" ] ; then  
168     ${WL_HOME}/common/bin/wlsifconfig.sh -addif "${IFNAME}" "${SERVER_IP}" "${IPMASK}"  
169 fi  
170  
# START WEBLOGIC  
172  
173 echo "starting weblogic with Java version:"  
174  
175 ${JAVA_HOME}/bin/java ${JAVA_VM} -version  
176  
177 if [ "${WLS_REDIRECT_LOG}" = "" ] ; then  
178     echo "Starting WLS with line:"  
179     echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS} -Dweblogic.Name=  
180     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS} -Dweblogic.Name=  
181 else  
182     echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"  
183     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS} -Dweblogic.Name=  
184 fi  
185  
186 stopAll  
187  
188 popd  
189 #  
189 export JAVA_OPTS="${JAVA_OPTS} -javaagent:E/tl/AppServerAgent/javaagent.jar"  
190 # Exit this script only if we have been told to exit.  
191  
192 if [ "${doExitFlag}" = "true" ] ; then  
193     exit  
194 fi
```

2b. Sample WebLogic v10.x startWebLogic.sh file

```

startWebLogic.sh
167 if [ "${SERVER_IP}" != "" ] ; then
168     ${WL_HOME}/common/bin/wlsifconfig.sh -addif "${IFNAME}" "${SERVER_IP}" "${IPMASK}"
169 fi
170
171 # START WEBLOGIC
172
173 echo "starting weblogic with Java version:"
174
175 ${JAVA_HOME}/bin/java ${JAVA_VM} -version
176
177 if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
178     echo "Starting WLS with line:"
179     echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS} -Dweblogic.Name=${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS} -Dweblogic.Name=${}
180 else
181     echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
182     ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS} -Dweblogic.Name=${}
183 fi
184
185 stopAll
186
187 popd
188
189 export JAVA_OPTS="$JAVA_OPTS -javaagent:B/tl/AppServerAgent/javaagent.jar"
190 # Exit this script only if we have been told to exit.
191
192 if [ "${doExitFlag}" = "true" ] ; then
193     exit
194 fi

```

3. Restart the application server. The application server must be restarted for the changes to take effect.

## OSGi Infrastructure Configuration

- Configuring OSGi Containers
  - To configure Equinox
  - To configure Apache Sling
  - To configure Felix for GlassFish
    - For GlassFish 3.1.2
  - To configure Felix for Jira or Confluence
    - Jira 5.x and newer
  - To configure other OSGi-based containers

### Configuring OSGi Containers

The GlassFish application server versions 3.x and later uses OSGi architecture. By default, OSGi containers follow a specific model for bootstrap class delegation. Classes that are not specified in the container's CLASSPATH are not delegated to the bootstrap classloader; therefore you must configure the OSGi containers for the App Server Agent classes.

For more information see [GlassFish OSGi Configuration per Domain](#).

To ensure that the OSGi container identifies the agent, specify the following package prefix:

org.osgi.framework.bootdelegation=com.singularity.\*

This prefix follows the regular boot delegation model so that the App Server Agent classes are visible.

If you already have existing boot delegations, add "com.singularity.\*" to the existing path separated by a comma. For example:

org.osgi.framework.bootdelegation=com.sun.btrace., com.singularity.

## To configure Equinox

1. Open the config.ini file located at <glassfish-install>/glassfish/osgi/equinox/configuration.
2. Add following package prefix to the config.ini file:

```
org.osgi.framework.bootdelegation=com.singularity.*
```

For more information see [Getting Started with Equinox](#).

## To configure Apache Sling

1. Open the sling.properties file. The location of the sling.properties varies depending on the Java platform. In the Sun/Oracle implementation, the sling.properties file is located at <java.home>/lib.

2. Add following package prefix to the sling.properties file.

```
org.osgi.framework.bootdelegation=com.singularity.*
```

## To configure Felix for GlassFish

1. Open the config.properties file, located at <glassfish-install>/glassfish/osgi/felix/conf.
2. Add following package prefix to the config.properties file.

```
org.osgi.framework.bootdelegation=com.singularity.*
```

## For GlassFish 3.1.2

Add:

```
com.singularity.*
```

to the boot delegation list in the <GlassFish\_Home\_Directory>\glassfish\config\osgi.properties file. For example:

```
org.osgi.framework.bootdelegation=${eclipselink.bootdelegation}, com.sun.btrace,
com.singularity.*
```

## To configure Felix for Jira or Confluence

1. From <atlassian-install>/bin (Stand-alone) or <Tomcat-home>/bin (EAR-WAR installation), open:
  - For Linux: setenv.sh
  - For Windows: setenv.bat
2. Update the Java options:
  - For Linux: JAVA\_OPTS=

- For Windows: set JAVA\_OPTS=%JAVA\_OPTS%

```
-javaagent:<path>/javaagent.jar
-Datlassian.org.osgi.framework.bootdelegation=<other_services>,com.singularity,
com.singularity.*,<other_services>
```

## Jira 5.x and newer

```
-Datlassian.org.osgi.framework.bootdelegation=META-INF.services,com.yourkit,com.yo
```

## To configure other OSGi-based containers

For other OSGi-based runtime containers, add the following package prefix to the appropriate OSGi configuration.

```
file.org.osgi.framework.bootdelegation=com.singularity.*
```

## Resin Startup Settings

- [To Configure Resin 1.x - 3.x](#)
  - To add the javaagent command in a Windows environment
  - To add the javaagent command in a Linux environment
- [To Configure Resin 4.x](#)

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to the resin.sh or resin.bat file.

## To Configure Resin 1.x - 3.x

### To add the javaagent command in a Windows environment

1. Open the resin.bat file, located at <Resin\_installation\_directory>/bin.
2. Add following javaagent argument to the beginning of your application server start script.

```
exec JAVA_EXE -javaagent:"<drive>:\<agent_home>\javaagent.jar"
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

```
-javaagent:"<drive>:\<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>"
```

The javaagent argument references the full path of the App Server Agent installation directory, including the drive.

3. Restart the application server for changes to take effect.

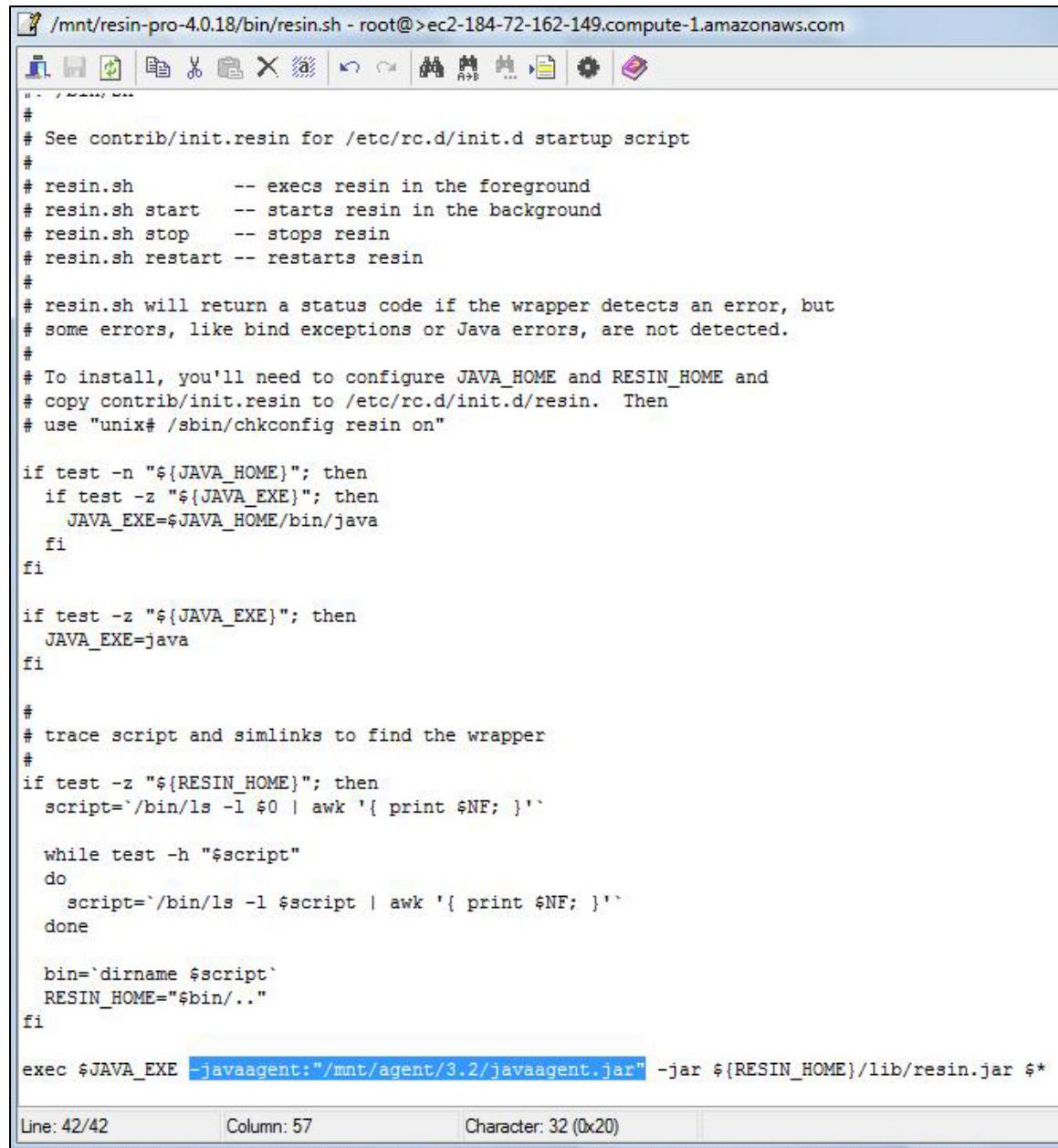
### To add the javaagent command in a Linux environment

1. Open the resin.sh file, located at <Resin\_Installation\_Directory>/bin.
2. Add the following javaagent argument to the beginning of your application server start script.

```
exec $JAVA_EXE -javaagent:"<agent_home>/javaagent.jar"
```

**⚠️** If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

The javaagent argument references the full path of the App Server Agent installation directory. See the following screenshot.



```
/mnt/resin-pro-4.0.18/bin/resin.sh - root@ec2-184-72-162-149.compute-1.amazonaws.com
# See contrib/init.resin for /etc/rc.d/init.d startup script
#
# resin.sh      -- execs resin in the foreground
# resin.sh start -- starts resin in the background
# resin.sh stop   -- stops resin
# resin.sh restart -- restarts resin
#
# resin.sh will return a status code if the wrapper detects an error, but
# some errors, like bind exceptions or Java errors, are not detected.
#
# To install, you'll need to configure JAVA_HOME and RESIN_HOME and
# copy contrib/init.resin to /etc/rc.d/init.d/resin. Then
# use "unix# /sbin/chkconfig resin on"

if test -n "${JAVA_HOME}"; then
  if test -z "${JAVA_EXE}"; then
    JAVA_EXE=${JAVA_HOME}/bin/java
  fi
fi

if test -z "${JAVA_EXE}"; then
  JAVA_EXE=java
fi

#
# trace script and simlinks to find the wrapper
#
if test -z "${RESIN_HOME}"; then
  script='`/bin/ls -l $0 | awk '{ print $NF; }`'

  while test -h "$script"
  do
    script='`/bin/ls -l $script | awk '{ print $NF; }`'
  done

  bin=`dirname $script`
  RESIN_HOME="$bin/.."
fi

exec $JAVA_EXE -javaagent:"/mnt/agent/3.2/javaagent.jar" -jar ${RESIN_HOME}/lib/resin.jar $*
```

Line: 42/42      Column: 57      Character: 32 (0x20)

3. Restart the application server. The application server must be restarted for the changes to take effect.

## To Configure Resin 4.x

1. To install the App Server Agent into Resin 4.X or later, edit the ./conf/resin.xml file and add:

```
<jvm-arg>-Xmx512m</jvm-arg>
<jvm-arg>-javaagent:<$appagent_location>/javaagent.jar</jvm-arg>
```

2. Restart the application server. The application server must be restarted for the changes to take effect.

## Solr Startup Settings

- To add the javaagent command in a Windows environment
- To add the javaagent command in a Linux environment

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to your Solr server.

### To add the javaagent command in a Windows environment

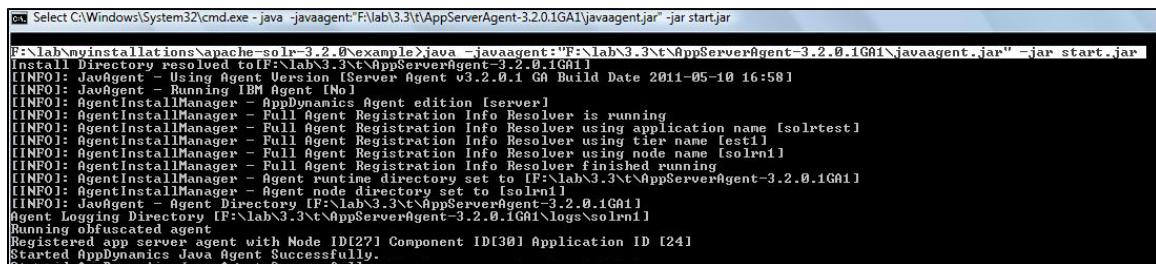
1. Open the Windows command line utility.
2. Execute the following commands to add the javaagent argument to the Solr server:

```
>cd $Solr_Installation_Directory
>java -javaagent:"<drive>:\<agent_home>\javaagent.jar" -jar start.jar
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

```
-javaagent:"<drive>:\<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>"
```

The javaagent argument references the full path to the App Server Agent installation directory, including the drive. For details see the screenshots.



A screenshot of a Windows Command Prompt window. The command entered is: Select C:\Windows\System32\cmd.exe - java -javaagent:"F:\lab\3.3\t\ApacheSolr\solr-4.2.1\bin\javaagent.jar" -jar start.jar. The output shows the Java Agent being loaded and the Solr server starting up successfully.

### To add the javaagent command in a Linux environment

1. Open the terminal.
2. Execute the following commands to add the javaagent argument to the Solr server:

```
>cd $Solr_Installation_Directory
>java -javaagent:"<agent_home>/javaagent.jar" -jar start.jar
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

The javaagent argument references the full path to the App Server Agent installation directory. For details see the screenshot.

```
root@domU-12-31-39-05-75-42:/mnt/solr/apache-solr-3.2.0/example
root@domU-12-31-39-05-75-42:/mnt/solr/apache-solr-3.2.0# ls
CHANGES.txt LICENSE.txt NOTICE.txt client contrib dist docs example
root@domU-12-31-39-05-75-42:/mnt/solr/apache-solr-3.2.0# cd example/
root@domU-12-31-39-05-75-42:/mnt/solr/apache-solr-3.2.0/example# java -javaagent:"/mnt/agent/3.2/test/javaagent.jar" -jar start.jar
Install Directory resolved to [/mnt/agent/3.2/test]
[INFO]: JavAgent - Using Agent Version [Server Agent v3.2.1.0 GA Build Date 2011-06-03 20:53]
[INFO]: JavAgent - Running IBM Agent [No]
[INFO]: AgentInstallManager - AppDynamics Agent edition [server]
[INFO]: AgentInstallManager - Full Agent Registration Info Resolver is running
[INFO]: AgentInstallManager - Full Agent Registration Info Resolver using application name [casstest]
[INFO]: AgentInstallManager - Full Agent Registration Info Resolver using tier name [test1]
[INFO]: AgentInstallManager - Full Agent Registration Info Resolver using node name [nodextestcassandra]
[INFO]: AgentInstallManager - Full Agent Registration Info Resolver finished running
[INFO]: AgentInstallManager - Agent runtime directory set to [/mnt/agent/3.2/test]
[INFO]: AgentInstallManager - Agent node directory set to [nodextestcassandra]
[INFO]: JavAgent - Agent Directory [/mnt/agent/3.2/test]
Agent Logging Directory [/mnt/agent/3.2/test/logs/nodextestcassandra]
Running obfuscated agent
Registered app server agent with Node ID[28] Component ID[31] Application ID [25]
Started AppDynamics Java Agent Successfully.
2011-06-08 12:24:06.068:INFO::Logging to STDERR via org.mortbay.log.StdErrLog
2011-06-08 12:24:06.359:INFO::jetty-6.1-SNAPSHOT
```

## Standalone JVM Startup Settings

- To add the javaagent command in a Windows environment
- To add the javaagent command in a Linux environment

AppDynamics works just as well with JVMs that are not application servers or containers.

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option, a standard Java option and can be used with any JVM. Add this option to your standalone JVM.

### To add the javaagent command in a Windows environment

1. Open the command line utility for Windows.
2. Add javaagent argument to the standalone JVM:

```
>java -javaagent:"Drive:\agent_home\javaagent.jar"
<fully_qualified_class_name_with_main_method>
```

For example:

```
>java -javaagent:"C:\AppDynamics\agentDir\javaagent.jar" com.main.HelloWorld
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

```
-javaagent : "<drive>:\<agent_home>\javaagent.jar=uniqueID=<my-app-jvm1>"
```

The javaagent argument references the full path to the App Server Agent installation directory, including the drive.

### To add the javaagent command in a Linux environment

1. Open the terminal.
2. Add the javaagent argument to the standalone JVM:

```
>java -javaagent:"/agent_install_dir/javaagent.jar"  
<fully_qualified_class_name_with_main_method>
```

For example:

```
>java -javaagent:"/mnt/AppDynamics/agentDir/javaagent.jar" com.main.HelloWorld
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

The javaagent argument references the full path to the App Server Agent installation directory.

## Tanuki Service Wrapper Configuration

- To configure the Tanuki Service Wrapper

The AppDynamics Java App Server Agent bootstraps using the javaagent command line option. Add this option to the Tanuki Service wrapper.conf file.

### To configure the Tanuki Service Wrapper

1. Open the wrapper.conf file.
2. Use the wrapper.java.additional.<n> property to add the javaagent option.

```
wrapper.java.additional.6=-javaagent:/C:/agent/javaagent.jar
```

 If you are a Self-Service Trial user, add the App Agent for Java javaagent argument to your JVM start script where <my-app-jvm1> is the name you use for the application running on that JVM.

For more information see:

- [Tanuki Service Wrapper Properties](#)
- [Example Configuration](#)
- [More Help On Tanuki Service Wrapper](#)

## Tibco BusinessWorks Configuration

There are typically two scripts associated with the Tibco BusinessWorks Services engine.

- my\_application.sh
- my\_application.tra

The JVM that runs the services start with a command line tool called bwengine(.exe).

Add the following to the .tra file:

```
Memory  
java.extended.properties -javaagent:/opt/appagent/javaagent.jar
```

See also: <https://ssl.tibcommunity.com/thread/14856>

## App Agent for Java Configuration Properties

- Where to Configure App Agent Properties
  - Creating and Registering Tiers
  - Example Java App Agent controller-info.xml File
  - Example Startup-up Using System Properties
  - Java App Server Agent Properties
    - Agent-Controller Communication Properties
      - Controller Host Property
      - Controller Port Property
    - Agent Identification Properties
      - Application Name Property
      - Tier Name Property
      - Node Name Property
    - Multi-Tenant Mode Properties
      - Account Name Property
      - Account Access Key Property
    - Proxy Properties for the Controller
      - Proxy Host Property
      - Proxy Port Property
    - Other Properties
      - Controller SSL Enabled Property
      - Enable Orchestration Property
      - Agent Runtime Directory Property
      - Redirect Logfiles Property
      - Force Agent Registration Property
      - Reuse Node Name Property
      - Auto Node Name Prefix Property
      - Chron/Batch JVM Property
      - Unique Host ID Property
  - Learn More

## Where to Configure App Agent Properties

You can configure the App Server Agent properties:

- in the controller-info.xml file in the <Agent\_Installation\_Directory>/conf directory
- in the system properties (-D options) in the JVM startup script

The system properties override the settings in the controller-info.xml file.

For shared binaries among multiple JVM instances, AppDynamics recommends using a combination of the xml file and the start-up properties to configure the app agent. Configure all the properties common to all the JVMs in the controller-info.xml file. Configure the properties unique to a JVM using the system properties in the start-up script.

For example:

- For multiple JVMs belonging to the same application serving different tiers, configure the application name in the controller-info.xml file and the tier name and node name using the system properties.
- For multiple JVMs belonging to the same application and the same tier, configure the application name and the tier name in the controller-info.xml file and the node name using the system properties.

After you configure agent properties, confirm that the javaagent argument has been added to the JVM startup script. For more information, see [Java Server-Specific Installation Settings](#).

For some properties, you can use system properties already defined in the start-up script as the App Server Agent property values. For more information, see [Configure App Agent for Java to Use Existing System Properties](#).

## Creating and Registering Tiers

You can create a tier in the Controller prior to setting up any agents. Alternatively, an agent can register its tier with the Controller the first time, and only the first time, that it connects with the Controller. If a tier with the name used to connect already exists, the agent is associated with the existing tier.

## Example Java App Agent controller-info.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<controller-info>

    <controller-host>192.168.1.20</controller-host>

    <controller-port>8090</controller-port>

    <controller-ssl-enabled>false</controller-ssl-enabled>

    <application-name>ACMEOnline</application-name>

    <tier-name>InventoryTier</tier-name>

    <node-name>Inventory1</node-name>

    <agent-runtime-dir></agent-runtime-dir>

    <enable-orchestration>false</enable-orchestration>

    <account-name></account-name>
    <account-access-key></account-access-key>

    <force-agent-registration>false</force-agent-registration>

</controller-info>
```

## Example Startup-up Using System Properties

The following command uses the system properties to start the agent that monitors the ACME Online sample application's Inventory tier.

```
java -javaagent:/home/appdynamics/AppServerAgent/
-DappDynamics.controller.hostName=192.168.1.20 -DappDynamics.controller.port=8090
-DappDynamics.agent.applicationName=ACMEOnline -DappDynamics.agent.tierName=Inventory
-DappDynamics.agent.nodeName=Inventory1 SampleApplication
```

## Java App Server Agent Properties

This section describes the Java App Agent configuration properties, including their controller-info-xml elements and their system property options.

## Agent-Controller Communication Properties

## Controller Host Property

**Description:** This is the host name or the IP address of the AppDynamics Controller. Example values are 192.168.1.22 or myhost or myhost.abc.com. This is the same host that you use to access the AppDynamics browser-based user interface. For an on-premise Controller, use the value for Application Server Host Name that was configured when the Controller was installed. If you are using the AppDynamics SaaS Controller service, see the Welcome email from AppDynamics.

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

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

**Type:** String

**Default:** None

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

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

## Controller Port Property

**Description:** This is the HTTP(S) port of the AppDynamics Controller. This is the same port that you use to access the AppDynamics browser-based user interface.

If the Controller SSL Enabled property is set to true, specify the HTTPS port of the Controller; otherwise specify the HTTP port. See [Controller SSL Enabled Property](#).

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

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

**Type:** Positive Integer

**Default:** For On-premise installations, port 8090 for HTTP and port 8181 for HTTPS are the defaults.  
For the SaaS Controller Service, port 80 for HTTP and port 443 for HTTPS are the defaults.

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

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

## Agent Identification Properties

### Application Name Property

**Description:** This is the name of the logical business application that this JVM node belongs to. Note that this is not the deployment name(ear/war/jar) on the application server.

If a business application of the configured name does not exist, it is created automatically.

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

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

**Type:** String

**Default:** None

**Required:** Yes

### Tier Name Property

**Description:** This is the name of the logical tier that this JVM node belongs to. Note that this is not the deployment name (ear/war/jar) on the application server.

If the JVM / AppServer start-up script already has a system property that references the tier, such as -Dserver.tier, you could use \${server.tier} as the tier name. For more information, see [Configure App Agent for Java to Use Existing System Properties](#).

See [Name Business Applications, Tiers, and Nodes](#).

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

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

**Type:** String

**Default:** None

**Required:** Yes

## Node Name Property

**Description:** This is the name of JVM node.

Where JVMs are dynamically created, use the system property to set the node name.

If your JVM / AppServer start-up script already has a system property that can be used as a node name, such as -Dserver.name, you could use \${server.name} as the node name. You could also use expressions such as \${server.name}\_\${host.name}.MyNode to define the node name. See [Configure App Agent for Java to Use Existing System Properties](#) for more information.

In general, the node name must be unique within the business application and physical host. If you want to use the same node name for multiple nodes on the same physical machine, create multiple virtual hosts using the Unique Host ID property. See [Unique Host ID Property](#).

See [Name Business Applications, Tiers, and Nodes](#).

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

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

**Type:** String

**Default:** None

**Required:** Yes

## Multi-Tenant Mode Properties

**Description:** If the AppDynamics Controller is running in multi-tenant mode or if you are using the AppDynamics SaaS Controller, specify the account name and account access key for this agent to authenticate with the Controller. If you are using the AppDynamics SaaS Controller, the account name is provided in the Welcome email sent by AppDynamics.

If the Controller is running in single-tenant mode (the default) there is no need to configure these values.

## Account Name Property

**Description:** This is the account name used to authenticate with the Controller.

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

**System Properties:** -Dappdynamics.agent.accountName

**Type:** String

**Default:** None

**Required:** Yes for AppDynamics SaaS Controller and other multi-tenant users; no for single-tenant users.

## Account Access Key Property

**Description:** This is the account access key used to authenticate with the Controller.

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

**System Properties:** -Dappdynamics.agent.accountAccessKey

**Type:** String

**Default:** None

**Required:** Yes for AppDynamics SaaS Controller and other multi-tenant users; no for single-tenant users.

## Proxy Properties for the Controller

These properties route data to the Controller through a proxy.

### Proxy Host Property

**Description:** This is the proxy host name or IP address.

**Element in controller-info.xml:** Not applicable

**System Property:** -Dappdynamics.http.proxyHost

**Type:** String

**Default:** None

**Required:** No

### Proxy Port Property

**Description:** This is the proxy HTTP(S) port.

**Element in controller-info.xml:** Not applicable

**System Property:** -Dappdynamics.http.proxyPort

**Type:** Positive Integer

**Default:** None

**Required:** No

## Other Properties

### Controller SSL Enabled Property

**Description:** When set to true, this property specifies that the agent should use SSL (HTTPS) to connect to the Controller. If SSL Enabled is true, set the Controller Port property to the HTTPS port of the Controller. See Controller Port Property.

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

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

**Type:** Boolean

**Default:** False

**Required:** No

### Enable Orchestration Property

**Description:** When set to true, enables auto-detection of the controller host and port when the app server is a compute cloud instance created by an AppDynamics orchestration workflow. See [Controller Host Property](#) and [Controller Port Property](#).

In a cloud compute environment, auto-detection is necessary for the Create Machine tasks in the workflow to run correctly.

If the host machine on which this agent resides is not created through AppDynamics workflow orchestration, this property should be set to false.

**Element in controller-info.xml:** <enable-orchestration>

**System Property:** Not applicable

**Type:** Boolean

**Default:** False

**Required:** No

## Agent Runtime Directory Property

**Description:** This property sets the runtime directory for all runtime files (logs, transaction configuration) for nodes that use this agent installation. If this property is specified, all agent logs are written to <Agent-Runtime-Directory>/logs/node-name and transaction configuration is written to the <Agent-Runtime-Directory>/conf/node-name directory.

**Element in controller-info.xml:** <agent-runtime-dir>

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

**Type:** String

**Default:** <Agent\_Installation\_Directory>/nodes

**Required:** No

## Redirect Logfiles Property

**Description:** This property sets the destination directory to which to redirect log files for a node.

**Element in controller-info.xml:** Not applicable

**System Property:** -Dappdynamics.agent.logs.dir

**Type:** String

**Default:** <Agent\_Installation\_Directory>/logs/<Node\_Name>

**Required:** No

## Force Agent Registration Property

**Description:** Set to true only under the following conditions:

- The Agent has been moved to a new application and/or tier from the UI and
- You want to override that move by specifying a new application name and/or tier name in the agent configuration.

**Element in controller-info.xml:** <force-agent-registration>

**System Property:** Not applicable

**Type:** Boolean

**Default:** False

**Required:** No

## Reuse Node Name Property

**Description:** Set this property if you want the Controller to generate unique node names automatically using a prefix.

You can specify the prefix in the [Auto Node Prefix Property](#). If you do not provide a prefix but set the reuse.nodeName property to true, the Controller uses the tier name as a prefix.

This property is useful for dynamic multi-tier clustered applications with many JVMs that have short life spans. It allows AppDynamics to reuse node names and to capture historical data for these short-lived nodes after they become historical or are deleted.

**Element in controller-info.xml:** Not applicable

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

**Type:** Boolean

**Default:** None

**Required:** No

## Auto Node Name Prefix Property

**Description:** Set this property if you want the Controller to generate node names automatically using a prefix that you provide.

The Controller generates node names based on the prefix concatenated with a number, which is incremented sequentially. For example, if you assign a value of "mynode" to this property, the Controller generates node names "mynode-1", "mynode-2" and so on.

If one of the nodes is deleted and the [Reuse Node Name Property](#) is true, the Controller will re-use the deleted node name.

**Element in controller-info.xml:** Not applicable

**System Property:** -Dappdynamics.agent.auto.node.prefix=<your\_prefix>

**Type:** String

**Default:** Serial number maintained by the Controller appended to the tier name

**Required:** No

## Chron/Batch JVM Property

**Description:** Set this property to true if the JVM is a batch/chron process or if you are instrumenting the main() method.

**Element in controller-info.xml:** Not applicable

**System Property:** -Dappdynamics.cron.vm

**Type:** Boolean

**Default:** False

**Required:** No

## Unique Host ID Property

**Description:** This property logically partitions a single physical host or virtual machine.

You can use the unique host id when you want to use the same node name for multiple nodes on the same physical machine.

Set the value to a string that is unique across the entire managed infrastructure. The string may not contain any spaces.

If this property is set on the app agent, it must be set on the machine agent as well.

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

**Type:** String

**Default:** None

**Required:** No

## Learn More

- Name Business Applications, Tiers, and Nodes
- Configure App Agent for Java for JVMs that are Dynamically Identified
- Configure App Agent for Java to Use Existing System Properties
- Java Agent on z-OS or Mainframe Environments Configuration

# Configure App Agent for Java for Batch Processes

- To configure the App Agent for Java
- To use the script name as the node name
- [Learn More](#)

You can configure the App Agent for Java for those JVMs that run as cron or batch jobs where the JVM runs only for the duration of the job. AppDynamics monitors the main method of the Java program.

## To configure the App Agent for Java

1. Add the application and tier name to the controller-info.xml file.
2. Add the appdynamics.cron.vm property to the AppDynamics javaagent command in the startup script of your JVM process:

```
-javaagent:<agent_install_dir>/javaagent.jar  
-Dappdynamics.agent.nodeName=${NODE_NAME} -Dappdynamics.cron.vm=true
```

The agent\_install\_dir is the full path of the App Agent for Java installation directory.

The appdynamics.cron.vm property creates a delay between the end of the main method and the JVM exit so that the Agent has time to upload metrics to the Controller.

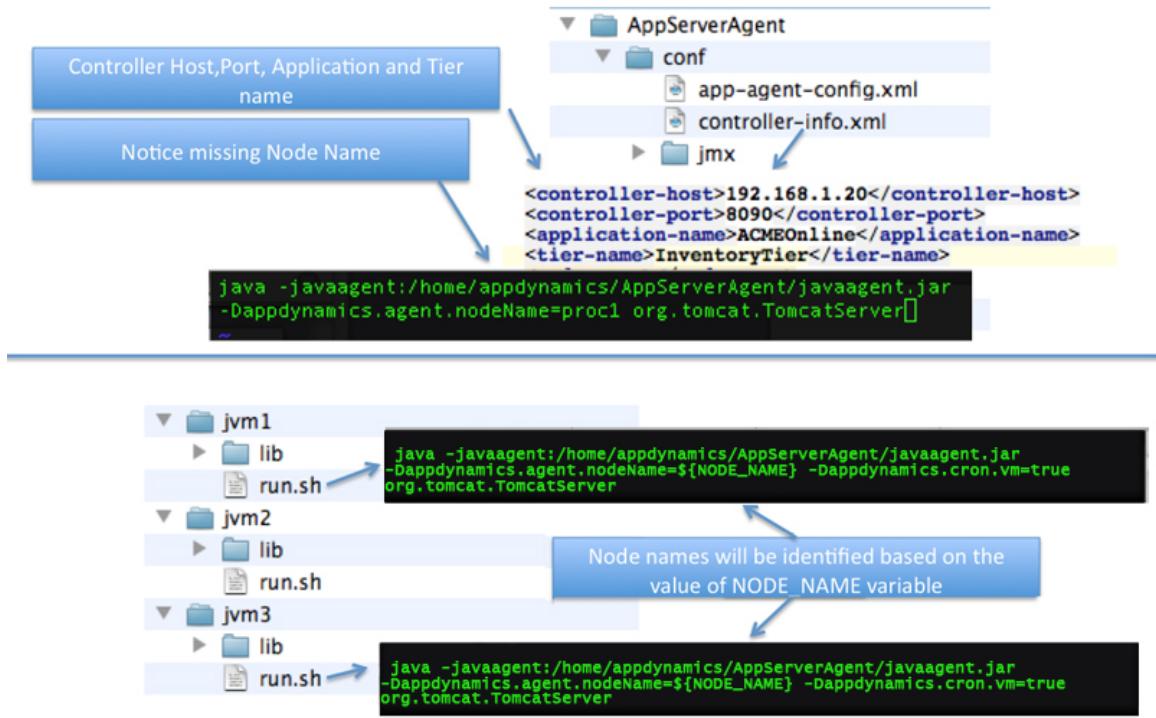
## To use the script name as the node name

You can use the script name that executes the cron or batch job as the node name.

The following commands set the value of variable NODE\_NAME using the combination of the script and host name. Add these commands to the startup script of the JVM.

```
# Use the name of the script (no path, no extension) as the name of the node.  
NODE_NAME=sample  
NODE_NAME="\${NODE_NAME%*.}"  
echo $NODE_NAME  
# Localize the script to the host.  
NODE_NAME="\$NODE_NAME@\${HOSTNAME}"
```

The following illustration shows the sample configuration for controller-info.xml and the startup script of the JVM.



## Learn More

- Configure Background Tasks (Java)

# Configure App Agent for Java for JVMs that are Dynamically Identified

- To configure the node name of the App Agent for Java
- Configuration notes

This topic describes how to configure the App Agent for Java in environments where the JVMs are dynamic.

### To configure the node name of the App Agent for Java

- Add the **application** and **tier name** to the controller-info.xml file.
- Add the javaagent argument and the following system properties (-D options) to the startup script of the JVMs:

```
java -javaagent:<agent-install-dir>/javaagent.jar -Dappdynamics.agent.nodeName=${NODE_NAME}
```

### Configuration notes

The system properties are separated by a white space character.

The <agent-install-dir> references the full path of the App Agent for Java installation directory.

The token \${NODE\_NAME} identifies the JVMs dynamically and names these JVMs based on the parameter value passed during the execution of the startup script for your JVM process.

The application and tier names can also be configured using the system properties. For more details see [App Agent for Java Configuration Properties](#).

Some application server management consoles allow you to specify startup arguments using a web interface. For details see [Java Server-Specific Installation Settings](#).

## Configure App Agent for Java in Restricted Environments

- To write the "startup hook" agent program

Some restricted environments do not allow any changes to the JVM startup script. For these environments AppDynamics provides the appdynamics.agent.startup.hook property. This "startup hook" allows a single point of deployment for the agent. You create a Java main method that is invoked programmatically, before your startup script is executed.

### To write the "startup hook" agent program

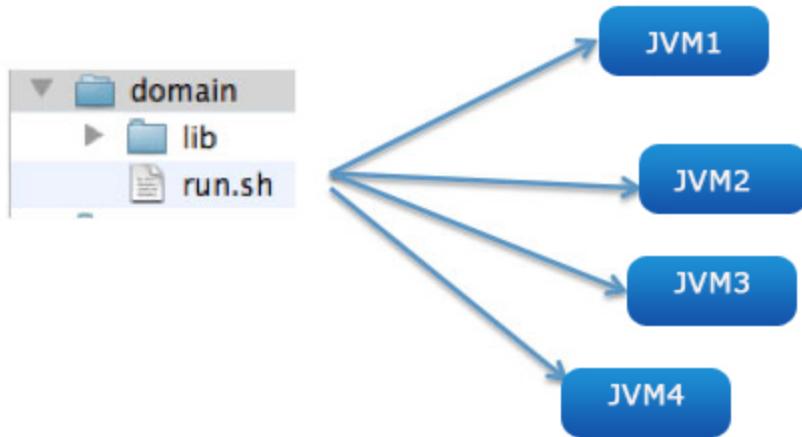
1. Implement a class with Java main method.
2. Create a JAR file for this class.
3. In the manifest of the JAR file, specify the class created in step 1.
4. Add the following javaagent argument and system properties (-D options) to your startup script:

```
-javaagent:<agent_install_dir>/javaagent.jar  
-Dappdynamics.agent.startup.hook=<JAR-file>
```

## Configure App Agent for Java in z-OS or Mainframe Environments

- To name nodes automatically
- To remove dead nodes
- Learn More

In some environments JVMs have transient identity, such as when a single script spawns multiple JVMs.



For example, an environment may consist of WebSphere on IBM Mainframes, using a dynamic workload management feature that spawns new JVMs for an existing application server (called a servant). These JVMs are exact clones of an existing JVM, but each of them has a different process ID. Based on load, any number of additional JVMs may be created.

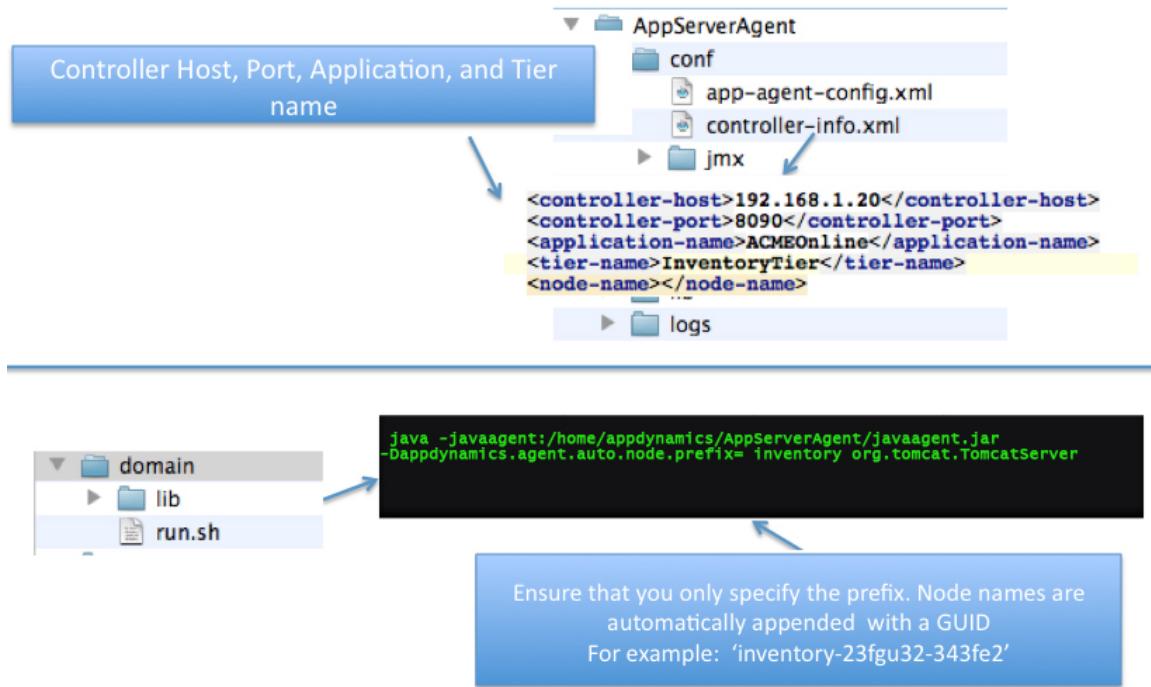
## To name nodes automatically

The App Server Agent can automatically name the dynamically generated JVMs using the `appdynamics.agent.auto.node.prefix` property. See [Reuse Node Name Property](#) to enable re-use of node names and [Auto Node Name Prefix Property](#) to set the prefix used for automatically-named nodes. You used these properties in your startup script.

```
-Dappdynamics.agent.auto.node.prefix=<node name prefix>
-Dappdynamics.agent.agent.reuse.nodeName=true
```

If you are using these properties, ensure that you have not specified the node name anywhere (controller-info.xml file or as a system property) in your JVMs start-up script.

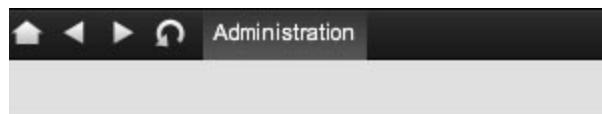
The following illustration shows the sample configuration for ACME Bookstore. This configuration will create unique node names for every instance of the virtual machine starting up in the ACME Bookstore environment.



## To remove dead nodes

In a typical environment, the nodes are recycled and new nodes get generated. AppDynamics strongly recommends that you remove the dead nodes both from the AppDynamics user interface (UI) as well as from the system.

1. Log in to the Controller administration console.
2. Go to "Controller Settings" section to see the advanced properties for the Controller.



## Accounts

Create and Manage Accounts

## Controller Settings

Configure the Controller

3. Set the retention and deletion properties, based on the requirements for your environment. AppDynamics recommends that you set the permanent deletion period at least an hour more than the retention period.

- **node.permanent.deletion.period:** Time (in hours) after which a node that has lost contact with the Controller is deleted permanently from the system.

- **node.retention.period:** Time (in hours) after which a node that has lost contact with the Controller is deleted. In this case, the AppDynamics UI will not display the node, however the system will continue to retain it.

metrics.write.thread.count	The count of parallel threads to be i	<input type="text" value="1"/>
multitenant.controller	Is the controller running in multi-tier	<input type="text" value="false"/>
node.permanent.deletion.period	Time (in hours) after which a node t	<input type="text" value="720"/>
node.retention.period	Time (in hours) after which a node t	<input type="text" value="500"/>

## Learn More

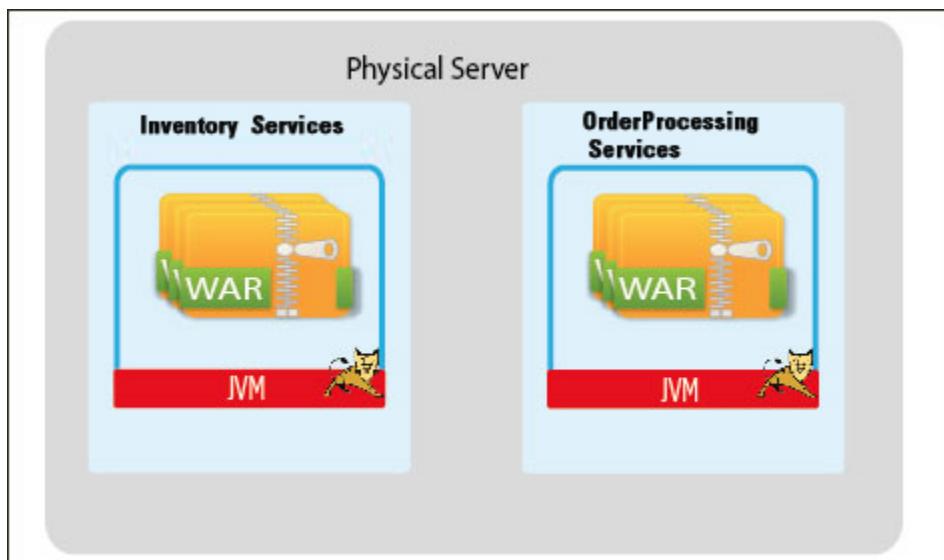
- [App Agent for Java Configuration Properties](#)

# Configure App Agent for Java on Multiple JVMs on the Same Machine that Serve Different Tiers

- [To configure the App Agent for Java](#)

This section describes how you can configure the App Agent for Java for multiple JVMs that are located on a single machine and are serving two tiers.

For example, the ACME Bookstore has two JVMs on the same physical server. These two JVMs are bound to two different virtual IP (one JVM is used for Order Processing Services and the other JVM is used for Inventory Services).



For such cases, follow these rules:

- All of the common information should be configured using controller-info.xml file.
- All of the information unique to a JVM should be configured using the system properties in the JVM startup script.
- Information in the startup scripts always overrides the information in the controller-info.xml file.

## To configure the App Agent for Java

1. Add application name to controller-info.xml file.

2. Add javaagent argument and following system properties (-D options) to the start-up script to each of your JVM:

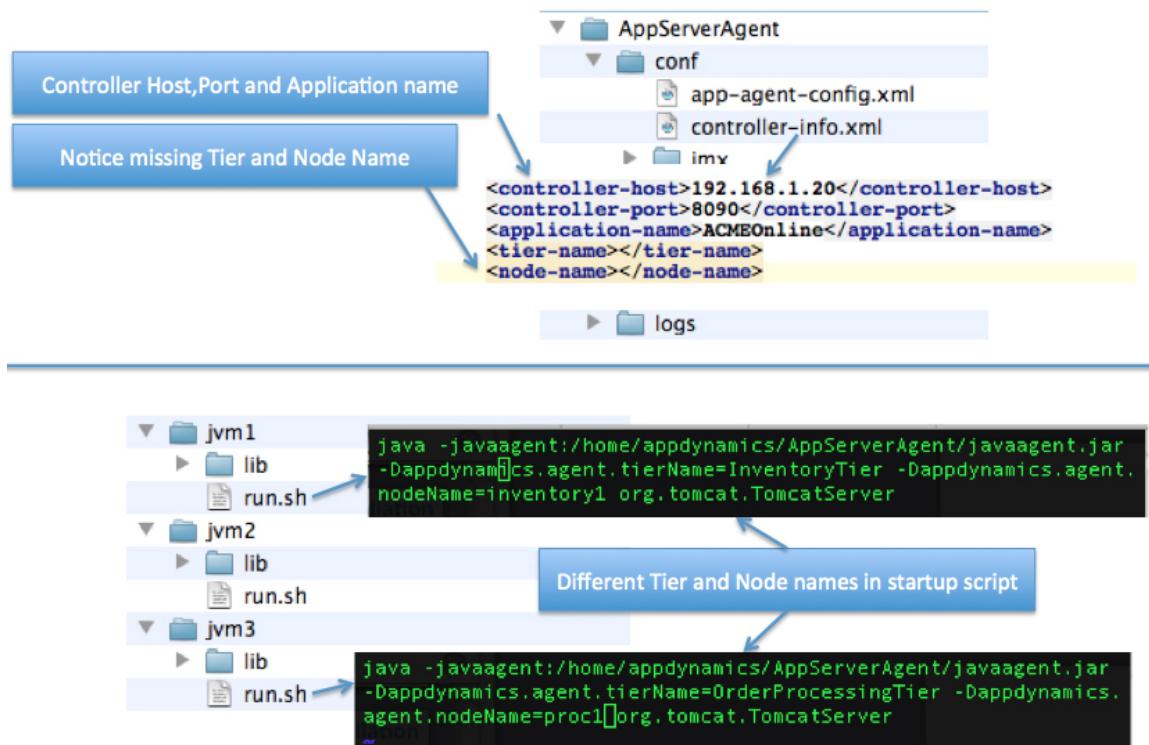
```
java -javaagent:<Agent-Installation-Directory>/javaagent.jar  
-Dappdynamics.agent.tierName=$tierName -Dappdynamics.agent.nodeName=$nodeName
```

Separate the system properties with a white space character.

All agents in shared mode need a unique node name so that they can be differentiated from one another. See [Configure App Agent for Java to Use Existing System Properties](#).

The following illustration displays how this configuration is applied to ACME Bookstore:

### Add Controller Host, Port, and Application Name in controller-info.xml file and add rest of the properties in the start-up script.



#### Tips:

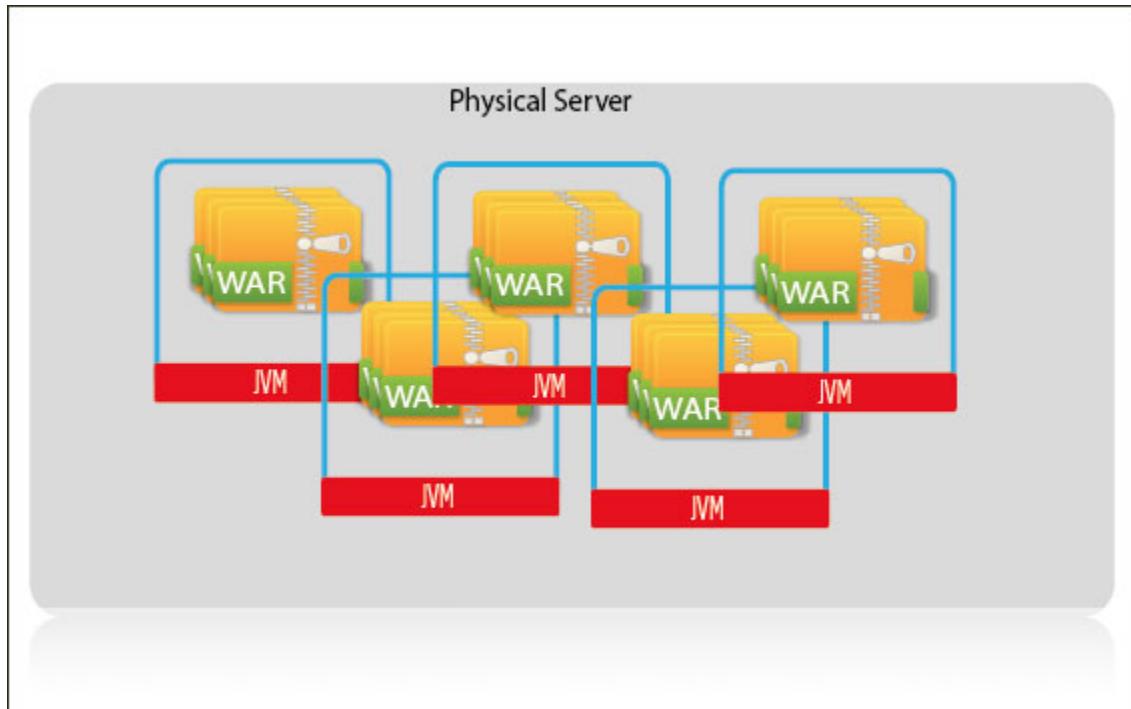
- The application and tier names can also be configured using the system properties. For more details see [App Agent for Java Configuration Properties](#).
- Some application server management consoles allow you to specify startup arguments using a web interface. See [Java Server-Specific Installation Settings](#).

## Configure App Agent for Java on Multiple JVMs on the Same Machine that Serves the Same Tier

- To configure the App Agent for Java properties

This topic describes how to configure the App Agent for Java for multiple JVMs that are located on a single machine and are serving the same tier.

For example, ACME Bookstore has a physical server with five JVMs installed on it. All of these JVMs are used for the Inventory Services.



For such cases, follow these rules:

- All of the common information should be configured using controller-info.xml.
- All of the information unique to a JVM should be configured using the system properties in the start-up script.
- Information in the startup scripts always overrides the information in the controller-info.xml file.

## To configure the App Agent for Java properties

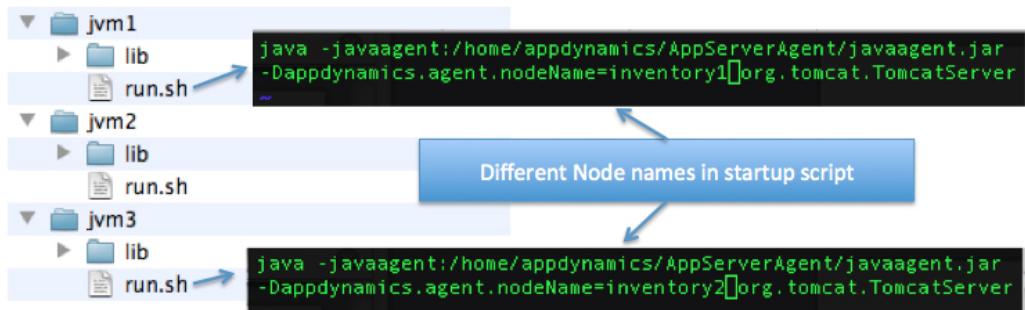
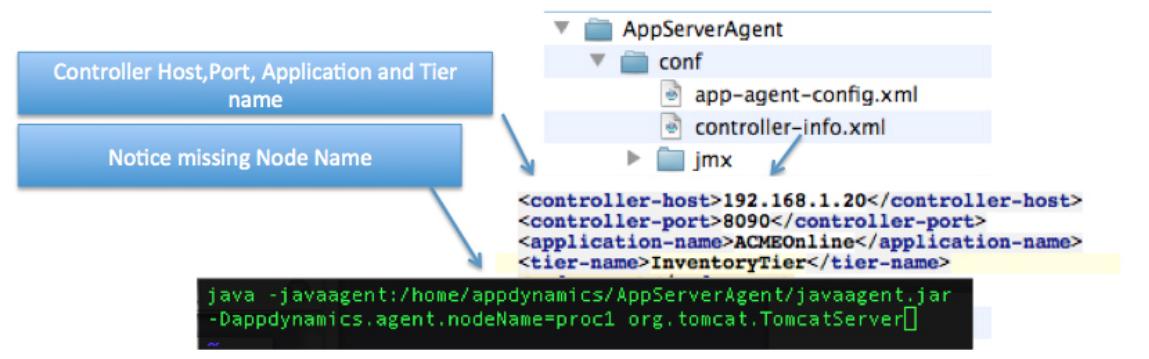
1. Provide the application and tier name in the controller-info.xml file.
2. Add the javaagent argument and system property (-D option) for the node name to the batch file or startup script of each JVM.

```
java -javaagent:<Agent-Installation-Directory>/javaagent.jar
-Dappdynamics.agent.nodeName=$nodeName
```

Separate the system properties with a white space character.

The following illustration displays how this configuration is applied to the ACME Bookstore.

Add Controller Host, Port, Application, and Tier Name in controller-info.xml file and add Node Name in the start-up script.



The application and tier names can also be configured using the system properties. See App Agent for Java Configuration Properties.

Some application server management consoles allow you to specify start-up arguments using a web interface. See Java Server-Specific Installation Settings.

## Configure App Agent for Java to Use Existing System Properties

- System Properties
  - Using System Properties
    - To identify nodes
    - To identify tiers
  - Sample Agent Configuration Using System Properties
  - Learn More

This topic explains how to configure the App Agent for Java using the existing system property values.

## System Properties

AppDynamics recommends that you use the existing system properties to configure the Agent when your environment consists of multiple JVMs on the same machine. Once you have these variables configured, you can complete Agent installation for all JVMs by simply adding the javaagent argument to each JVM startup script. Then add the rest of the information to the controller-info.xml file.

AppDynamics recommends that you use the system properties if the same startup script is starting all the JVMs in your environment.

You can identify the node name based on the value of -Dserver.name and the tier name based on the value of -Dcluster.name.

Also, you can combine two or more system properties to identify the node or tier name. You can use -Dhost.name and -Dserver.name to identify similarly named nodes on different machines even when they belong to the same tier.

## Using System Properties

Use the following syntax to represent the value of the system property in the controller-info.xml file.

```
${system.property.name}
```

You can combine multiple system properties.

```
${host.name}${server.name}
```

You can combine system properties with literals. In the following example '\_' and 'inventory' are literals.

```
${host.name}_${server.name}.inventory
```

### To identify nodes

```
${host.name}
```

or

```
${server.name}
```

or

```
${host.name}${server.name}
```

### To identify tiers

```
${cluster.name}
```

## Sample Agent Configuration Using System Properties

Consider a JVM with a script file named startserver.sh. This script file has following system property:

```
-Dserver.name=$1
```

If you execute:

```
startserver.sh ecommerce01
```

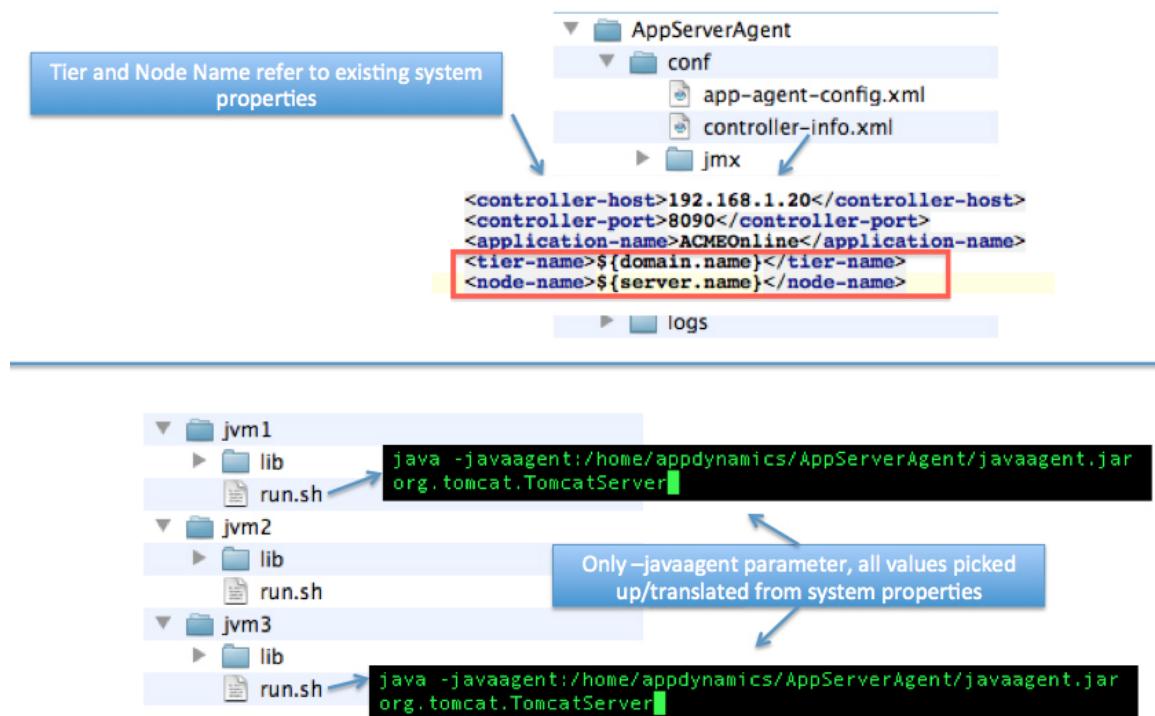
The script creates a new server named ecommerce01.

To use this system property for Agent configuration, add the appdynamics.nodeName property to startserver.sh file.

```
-Dappdynamics.nodeName=$server.name
```

When the script creates the new server named ecommerce01, it will be identified by AppDynamics as both a server and as a node.

The following screenshot shows a sample configuration for the controller-info.xml file and the startup script.



## Learn More

- Logical Model
- Install the App Agent for Java

# Configure AppDynamics for .NET

- Configure Custom Exit Points (.NET)
- Getter Chains in .NET Configurations
- Enable Thread Correlation (.NET)
- Configure the .NET Machine Agent
- Enable Correlation for .NET Remoting

## Configure Custom Exit Points (.NET)

- Default Backends Discovered by the Agent for .NET
- Configure Custom Exit Points for .NET Backends
- To create a custom exit point
  - To split an exit point
  - To group an exit point
- To define custom metrics for a custom exit point
- To define transaction snapshot data collected
- Learn More

AppDynamics provides default automatic discovery for commonly-used backends. If a backend used in your environment is not discovered, first compare the list of default backends to determine whether you need to modify the default configuration. If it is not on the list then configure a custom exit point according to these instructions.

## Default Backends Discovered by the Agent for .NET

The default data access backends:

- ADO.NET

The default remote services for .NET agents are:

- WCF
- HTTP
- Web Services, including SOAP
- Directory Services, including LDAP\*
- Queues
  - Apache ActiveMQ
  - IBM WebSphere MQ (also known as IBM XMS)
  - Microsoft Message Queuing ( MSMQ)\*
  - .NET Remoting\*
  - Tibco Enterprise Message Service (EMS)
  - Tibco Rendezvous (RV)
  - MicrosoftServiceBus (Windows Azure Service Bus)
  - MicrosoftServiceBusQueue (Windows Azure Service Bus Queues)
  - AzureQueue (Windows Azure Queues)

\* Notes: Correlation is supported unless otherwise indicated below.

- For LDAP, correlation is non-applicable
- For MSMQ, correlation for downstream calls is not supported.
- For .NET remoting, additional configuration is needed to get downstream correlation. See [Enable Correlation for .NET Remoting](#).

For instructions on modifying default backend discovery see [Configure Backend Detection](#).

## Configure Custom Exit Points for .NET Backends

Custom exit points provide identification for backend types that are not automatically detected, such as file systems, mainframes etc. For example, you can define a custom exit call to monitor the file system read method. Custom exit points appear as unresolved

backends in the flow maps. Unresolved backends are shown on flow maps with this icon .

You define a custom exit point by specifying the class and method used to identify the backend. If the method is overloaded, you need to add the parameters to identify the method uniquely.

You can restrict the method invocations for which you want AppDynamics to collect metrics by specifying match conditions for the method. The match conditions can be based on a parameter or the invoked object.

You can also optionally split the exit point based on a method parameter, the return value, or the invoked object.

You can also configure custom metrics and transaction snapshot data to collect for the backend.

See [Configurations for Custom Exit Points](#) for suggested custom configurations for some common backends.

## To create a custom exit point

1. In the left navigation panel, click **Configure -> Instrumentation**.
2. Click the **Backend Detection** tab.
3. Click the tab corresponding to the backend platform.
4. Select the application or tier for which you are configuring the custom exit point. Backend detection configuration is applied on a hierarchical inheritance model. See [Hierarchical Configuration Model](#).
5. Scroll down to Custom Exit Points.
6. Click **Add** (the + icon).
7. In the Create Custom Exit Point window, click the **Identification** tab if it is not selected.
8. Enter a name for the exit point. This is the name that identifies the backend.
9. Select the type of backend from the Type drop-down menu or check Use Custom if the type is not listed.
10. Configure the class and method name that identify the custom exit point.  
If the method is overloaded, check the Overloaded check box and add the parameters.
11. If you want to restrict metric collection based on a set of criteria that are evaluated at runtime, click **Add Match Condition** and define the match condition(s).  
For example, you may want to collect data only if the value of a specific method parameter contains a certain value.
12. Click **Save**.

The following screenshot shows a custom exit point of Type Cache. This exit point is defined on the `getAll()` method of the specified class. The exit point appears in flow maps as an unresolved backend named CoherenceGetAll.

**Edit Custom Exit Point**

Name:	CoherenceGetAll	Type:	Cache				
<input checked="" type="radio"/> Identification <input type="radio"/> Custom Metrics <input type="radio"/> Snapshot Data							
<b>Define the class and method name which, when called, will be identified as a Custom Exit Point</b>							
Class	that implements an Interface which	equals	com.tangosol.net.NamedCache				
Method Name	getAll	<input type="checkbox"/> Is this Method Overloaded?					
Method Parameters (optional)							
<input type="button" value="Add Parameter"/>							
Match Conditions (optional)							
<input type="button" value="Add Match Condition"/>							
<b>Calls to the specified class and method name can be further split based by a combination of match conditions.</b>							
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Cachename</td> <td>Collect data from the invoked object and capture the result.</td> </tr> </tbody> </table>		Name	Description	Cachename	Collect data from the invoked object and capture the result.	<input type="button" value="Add"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>	
Name	Description						
Cachename	Collect data from the invoked object and capture the result.						
<input type="button" value="Cancel"/> <input type="button" value="Save"/>							

## To split an exit point

1. Click **Add**.
2. Enter a display name for the split exit point.
3. Specify the source of the data (parameter, return value, or invoked object).
4. Specify the operation to invoke on the source of the data (`toString()` or getter chain for complex objects).
5. Click **Save**.

The following example shows a split configuration of the previously created CoherenceGetAll exit point based on the `getCacheName()` method of the invoked object.

**Edit Custom Exit Point Identifier**

Specify the parameter index or indicate if it the return value of the diagnostic data to be collected.  
Simple getters without parameters can be used on the parameter or the return value to be displayed against the display name specified here.

Display Name

*Create your own name for the data collected. This will be the display name for the data in Request Snapshots*

Collect Data From  Method Parameter @ Index:

Return Value

Invoked Object

Operation on Invoked Object  Use `toString()`

Use Getter Chain

*for example: `getAccount().getBalance()`*

## To group an exit point

You can group methods as a single exit point. The only requirement is that these methods point to the same key.

For example, ACME Online has an exit point for `NamedCache.getAll`. This exit point has a split configuration of `getCacheName()` on the invoked object as illustrated in the previous screenshot.

Suppose we also define another exit point for `NamedCache.entrySet`. This is another exit point, but it has the split configuration that has `getCacheName()` method of the invoked object.

**Edit Custom Exit Point**

Name:	CoherenceCacheAccess	Type:	Cache				
<input checked="" type="radio"/> Identification <input type="radio"/> Custom Metrics <input type="radio"/> Snapshot Data							
<b>Define the class and method name which, when called, will be identified as a Custom Exit Point:</b>							
Class	that implements an Interface which		equals com.tangosol.net.NamedCache				
Method Name	entrySet	<input type="checkbox"/> Is this Method Overloaded?					
Method Parameters (optional)							
<input type="button" value="Add Parameter"/>							
Match Conditions (optional)							
<input type="button" value="Add Match Condition"/>							
<b>Calls to the specified class and method name can be further split based by a combination of method parameters and match conditions.</b>							
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>CacheName</td> <td>Collect data from the invoked object and capture the result of getCacheName().</td> </tr> </tbody> </table>				Name	Description	CacheName	Collect data from the invoked object and capture the result of getCacheName().
Name	Description						
CacheName	Collect data from the invoked object and capture the result of getCacheName().						
<input type="button" value="Add"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>							

If the getAll() and the entrySet() methods point to the same cache name, they will point to the same backend.

Matching name-value pairs identify the back-end. In this case, there is only one key, i.e. cache name, has to be matched. So, here both exit points have the same name for the cache and they resolve to the same backend.

## To define custom metrics for a custom exit point

Custom metrics are collected in addition to the standard metrics.

The result of the data collected from the method invocation must be an integer value, which will either be averaged or added per minute, depending on your selection of data roll-up.

To configure custom business metrics that can be generated from the Java method invocation:

1. Click the **Custom Metrics** tab.
2. Click **Add**.
3. In the Add Custom Metric window, enter a name for the metric.
4. Select the Collect Data From radio button to specify the source of the metric data.
5. Select the Operation on Method Parameter to specify how the metric data is processed.
6. Select how the data should be rolled up (average or sum) from the Data Rollup drop-down menu.
7. Click **Create Custom Metric**.

## To define transaction snapshot data collected

1. Click the **Snapshot Data** tab.
2. Click **Add**.
3. In the Add Snapshot Data window, enter a display name for the snapshot data.
4. Select the Collect Data From radio button to specify the source of the snapshot data.
5. Select the Operation on Method Parameter to specify how the snapshot data is processed.
5. Click **Save**.

## Learn More

- Configurations for Custom Exit Points
- Monitor Remote Services
- Monitor Databases

## Getter Chains in .NET Configurations

- Property Getter Chains in .NET Configurations
- Learn More

This topic provides some guidance and examples of the correct syntax for getter chains in AppDynamics configurations.

## Property Getter Chains in .NET Configurations

Support in .NET configurations is limited to property getter chains. Only simple properties and fields can be chained.

Use the syntax:

<Property>.<Value>.<Member>

as in

```
Request.Url
```

or

```
Amount.Total
```

Getter chains of methods and array elements are not supported for .NET.

## Learn More

- Configure Business Transaction Detection
- Configure Data Collectors

## Enable Thread Correlation (.NET)

- To enable thread correlation for .NET applications
- Learn More

This topic describes how to enable thread correlation for .NET applications.

## To enable thread correlation for .NET applications

AppDynamics supports instrumentation for Thread.Start and ThreadPool.QueueUserWorkItem on the Common Language Runtime (CLR) 2.x and CLR 4.x.

1. Edit the application.config file and uncomment the relevant code:

```
<instrumentation>
    <!-- Uncomment the below to enable thread correlation -->
    <!--instrumentor name="ThreadCorrelationThreadPoolCLR2Instrumentor"
disable="false"/-->
    <!--instrumentor name="ThreadCorrelationThreadPoolCLR4Instrumentor"
disable="false"/-->
    <!--instrumentor name="ThreadStartCLR2Instrumentor" disable="false"/-->
    <!--instrumentor name="ThreadStartCLR4Instrumentor" disable="false"/-->
</instrumentation>
```

2. Save the file.
3. Restart the Coordinator and relevant CLR processes.

## Learn More

- [Configure Backend Detection](#)
- [Monitor Remote Services](#)

## Configure the .NET Machine Agent

- [The .NET Machine Agent](#)
  - To configure the .NET Machine Agent
- [The Standalone Machine Agent](#)
  - To Download and Install the Standalone Machine Agent
  - To Configure the Standalone Machine Agent to Run Automatically on Startup
- [Learn More](#)

## The .NET Machine Agent

There is a Machine Agent embedded in AppDynamic's App Agent for .NET, which is automatically installed with the app agent. You must configure the .NET Machine Agent before it can capture metrics in your managed environment.

The .NET Machine Agent does not support the extensions (plugins, metric listener, orchestration) that are available in the Standalone Machine Agent (which is a Java application).

You configure the .NET Machine Agent using the AppDynamics Agent Coordinator (the Coordinator) that also ships with the App Agent for .NET. The Coordinator determines the instrumentation and specifies the metrics captured by the Machine Agent.

## To configure the .NET Machine Agent

1. Open the application.config file located at <agent\_install\_dir>\Machine Agent\Configuration.
2. Specify how the machine agent connects to the Controller by modifying the <controller-host> and <controller-port> properties in the application.config file.

Property Name	Required	Default
---------------	----------	---------

<b>Controller Host</b> <i>For on-premise Controller installation:</i> Provide value as configured for "Application Server Host Name". <i>For SaaS Controller service, refer to the Welcome mail.</i>	<controller-host>	Yes	None
<b>Controller Port</b> <i>For on-premise Controller installation:</i> Provide value for "HTTP Listener Port". <i>For SaaS Controller service, refer to the Welcome mail.</i>	<controller-port>	Yes	<b>For On-premise installations:</b> Port 8090 is the default for HTTP and port 8181 is the default for HTTPS <b>For SaaS Controller Service:</b> Port 80 for HTTP and port 443 for HTTPS

For example:

```
<AppDynamicsAgentCoordinator>
  <controller-info>
    <controller-host>localhost</controller-host>
    <controller-port>8090</controller-port>
    ...
  </controller-info>
</AppDynamicsAgentCoordinator>
```

### 3. Specify how the Machine Agent should identify the Business Application.

The machine agent requires the name of its business application and tier. For more information see [Logical Model](#).

Property Name		Required	Default
<b>Application Name</b> <i>Name of the business application.</i>	<application-name>	Yes	None
<b>Tier Name</b> <i>Name of the tier</i>	<tier-name>	Required only if no app agent is deployed on the machine.	None

For example:

```
<AppDynamicsAgentCoordinator>
  <controller-info>
    .....
  <application-name>ACMEOnline</application-name>
  <tier-name>InventoryTier</tier-name>
  .....
</controller-info>
</AppDynamicsAgentCoordinator>
```

## The Standalone Machine Agent

If you want to monitor hardware metrics for an application that is running on Windows or use extensions such as the HTTP Metric Listener, or automate a local script as part of a policy, install and configure the [standalone Machine Agent](#). The standalone Machine Agent, which is a Java application, is different from the .NET Machine Agent. The Standalone Machine Agent provides the capability to use extensions (plugins, metric listener, orchestration) that are missing from the embedded .NET Machine Agent.

### To Download and Install the Standalone Machine Agent

- From the AppDynamics download site at <http://download.appdynamics.com/>, locate the AppDynamics Machine Agent.
- Click **Download Now**.

3. Unzip the MachineAgent.zip file.

## To Configure the Standalone Machine Agent to Run Automatically on Startup

1. Make sure that Java is installed on the machine.

2. Create a batch file containing the following line:

```
java -Xmx8m -jar <path to machineagent>/machineagent.jar
```

If your application loads a large number of extensions into memory, you may want need to increase the size of the memory allocation or drop it altogether:

```
java -jar <path to machineagent>/machineagent.jar
```

3. Click **Control panel -> Scheduled Tasks**.

4. Create a new task.

5. Select the batch file to execute. This is the file created in step 2.

6. Select **When my computer starts**.

7. Enter the administrator's credentials to run the Machine Agent as that user.

8. Click **Finish**.

To start the Machine Agent for the first time, right-click on the scheduled task, and click **Run**.

## Learn More

- [Install the Machine Agent](#)
- [Install the App Agent for .NET](#)
- [Monitor CLRs](#)

## Enable Correlation for .NET Remoting

Correlation for .NET Remoting is not enabled by default. This topic describes how to enable correlation for .NET Remoting.

### To enable correlation for .NET Remoting

1. Edit the application.config file and locate the <instrumentation> element. The XML looks similar to the following:

```
<instrumentation>
    <!-- Uncomment the below to enable thread correlation -->
    <!--instrumentor name="ThreadCorrelationThreadPoolCLR2Instrumentor"
disable="false"/-->
    <!--instrumentor name="ThreadCorrelationThreadPoolCLR4Instrumentor"
disable="false"/-->
    <!--instrumentor name="ThreadStartCLR2Instrumentor" disable="false"/-->
    <!--instrumentor name="ThreadStartCLR4Instrumentor" disable="false"/-->
</instrumentation>
```

2. Add lines these to the <instrumentation> element in machine agent configuration file:

```
<instrumentor name="RemotingEntryInstrumentor" disable="false">\>
<instrumentor name="RemotingExitInstrumentor" disable="false">\>
```

2. Save the file.
3. Restart the Coordinator and relevant CLR processes.

 For .NET Remoting, AppDynamics supports HTTP and Net.TCP protocols, but does not support the net.pipe protocol.

## Learn More

- Configure Backend Detection
- Monitor Remote Services

# Configure the App Agent for .NET

- Configuration Considerations
- Configuring an App Agent for .NET
  - To access the .NET Agent Configuration Utility
  - To set up the logs and account permissions
  - To provide Controller configuration information
  - To map business applications, tiers, and nodes to your application environment
  - To let AppDynamics automatically name the tiers
  - To manually name the tiers
- Using a Configuration File from the Command Line
  - To create a configuration file
  - To run the configuration utility from the command line
- Learn More

The App Agent for .NET requires information about your IIS and non-IIS applications on .NET. You configure the agent using the App Agent for .NET Configuration Utility.

Configure the App Agent for .NET according to what kind of application you want to monitor:

- For IIS applications, use the Configuration Utility with either automatic or manual tier naming. See the instructions in this topic.
- For non-IIS applications and services, set an environment variable and use the Configuration Utility with automatic tier naming. See [Enable the App Agent for .NET for Non-IIS Applications](#).
- For non-IIS applications and services, set an environment variable, use the Configuration Utility with manual tier naming, and manually edit the application configuration file. See [Enable the App Agent for .NET for Non-IIS Applications](#).

Use the App Agent for .NET Configuration Utility to configure the agent just after installation, or to make changes to existing agent configurations.

## Configuration Considerations

AppDynamics recommends that you install the Controller, or have access credentials to a SaaS Controller, before installing an agent.

The utility configures one agent at a time.

AppDynamics implements the profile API of the .NET CLR. Since only one profiler can be used at a time on a Windows machine, you may need to uninstall any pre-existing profiler, such as Ant, VS 2010 Performance Tools, or others. The utility will alert you if it finds a pre-existing profiler.

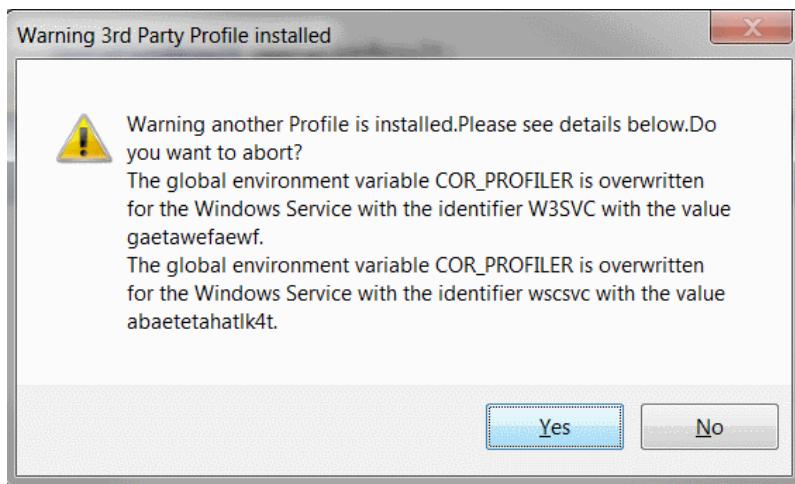
 To apply configurations, the .NET Agent Configuration Utility must restart IIS.

Prior to configuration, run the .NET Agent Installer. See [Install the App Agent for .NET](#).

# Configuring an App Agent for .NET

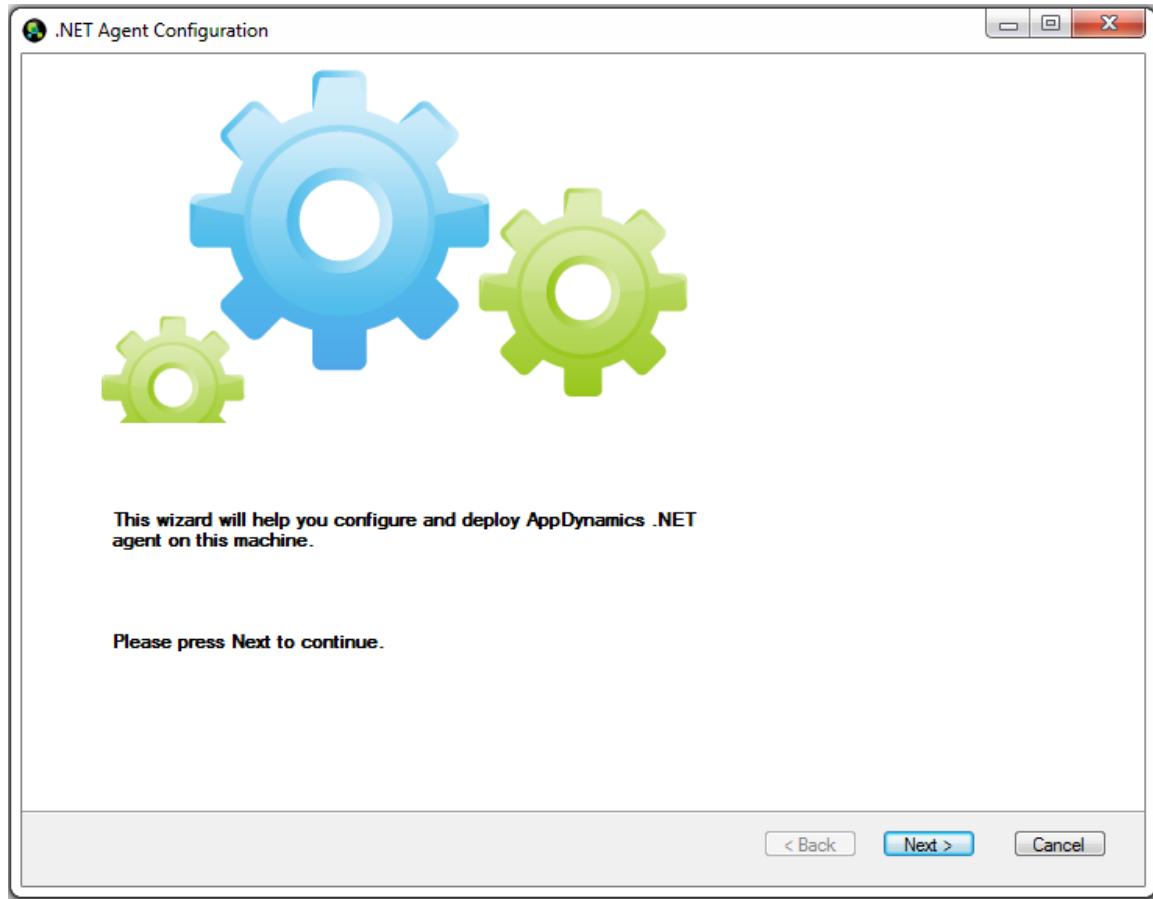
## To access the .NET Agent Configuration Utility

1. In the Windows menu, click **AppDynamics -> .NET Agent Configuration**.
2. If you get a message "Warning: 3rd Party Profiler installed" that means that another profiler is already installed. There can be only one profiler per Windows machine, and since AppDynamics uses a profiler you must uninstall any other profilers.



Click **OK** to exit and uninstall any pre-existing profiler. Check the registry to make sure that the uninstall process cleaned up the registry entries. Use the warning message to identify any undeleted profiler environment variables.

3. When there are no profiler conflicts, the Welcome screen appears.

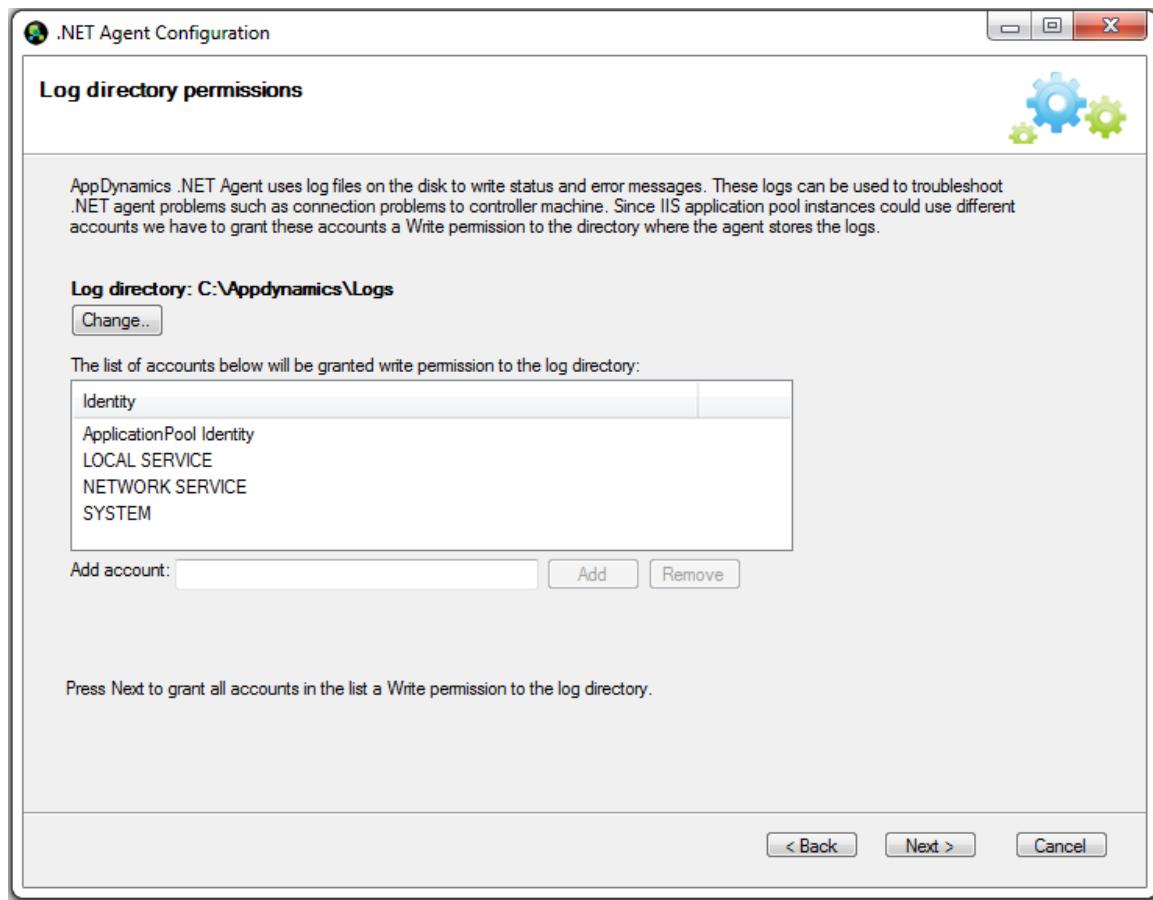


Click **Next**. The Log directory permissions window appears.

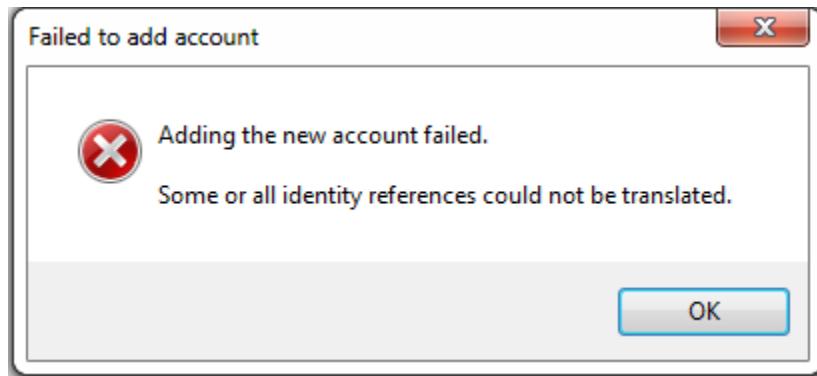
## To set up the logs and account permissions

The first screen helps you set up the location of the agent logs and provide the correct account access to the logs.

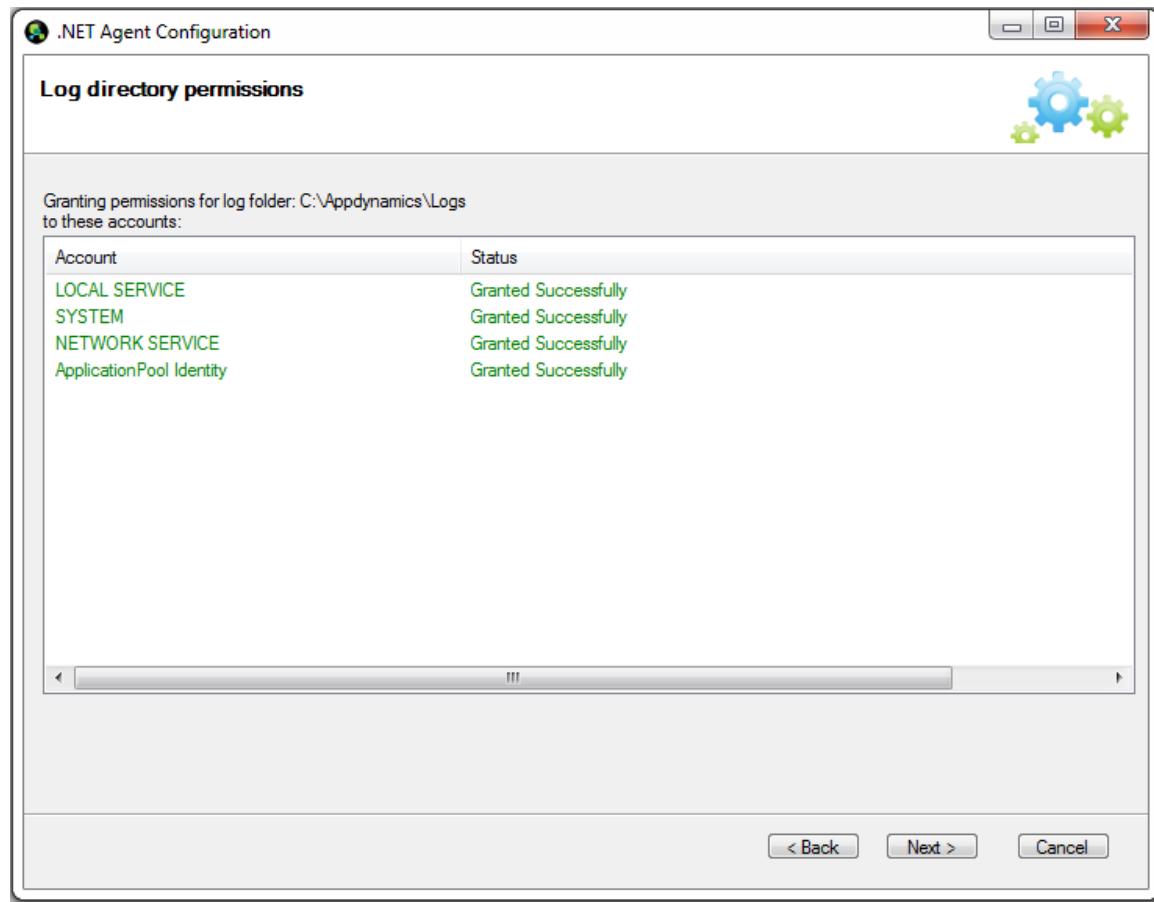
1. If you want to change the default location of the log directory, click **Change** and select a new location.
2. If needed, add accounts for log directory permissions.  
Commonly-used accounts are provided. If your application uses another account, enter the Windows account name that is used by the application and application pools to be monitored. The account name must be valid on the operating system and have permission to write to the log files directory.



Click **Add**. If you get a warning message, make sure that the account is valid on the system.



3. Click **Next** and the wizard confirms the list of accounts.



Click **Next**. The Controller Configuration window appears.

## To provide Controller configuration information

Enter the Controller access information and credentials.

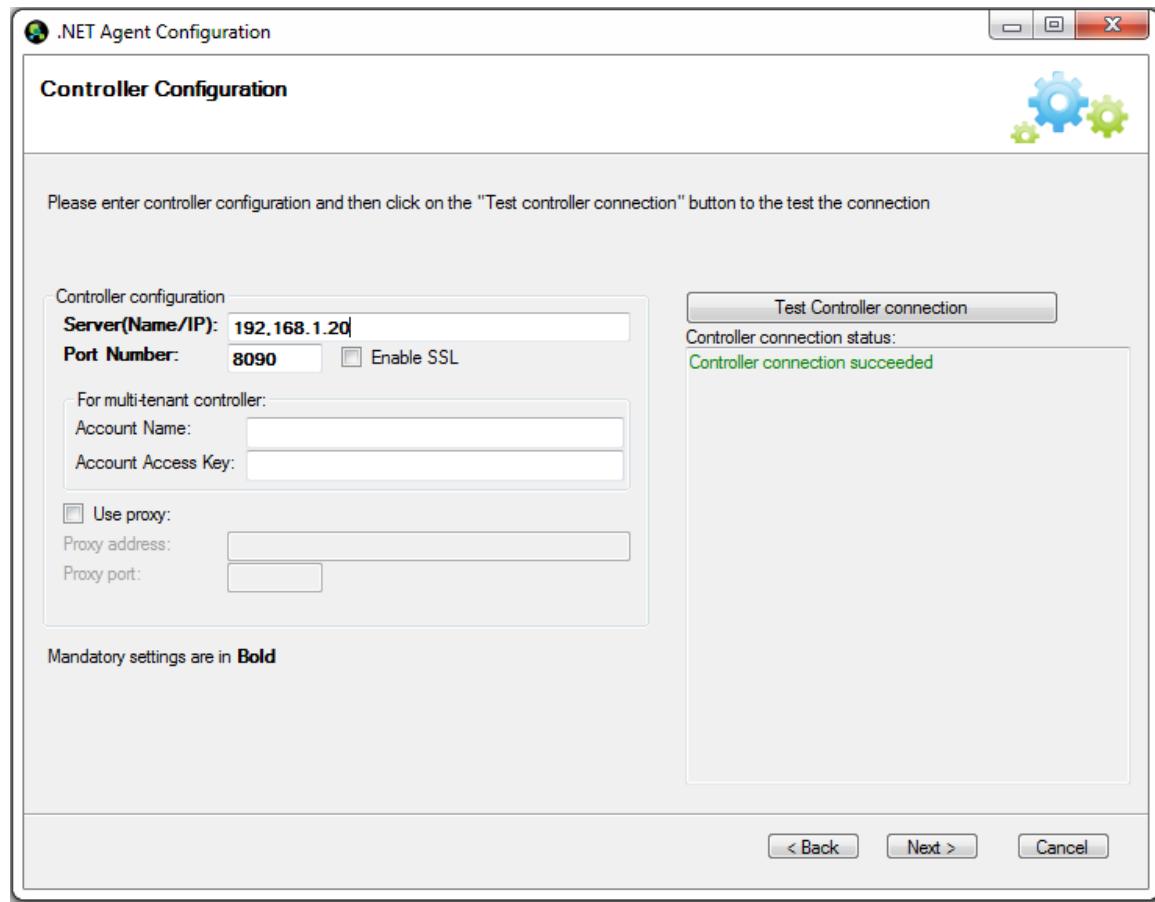
- For a SaaS Controller, enter the server name or IP, port number, account name, and access key as provided to you by AppDynamics.
- For an on-premise Controller, if you haven't already installed it, cancel this installation and see [Install the Controller](#). Otherwise enter the server name and port number of an existing Controller.
- For a secure connection, click **Enable SSL**.



Note: The Controller must use a trusted certificate.

- If needed, fill in the proxy information. AppDynamics does not support proxies that require authentication out-of-the-box.

5. Click **Test Controller Connection** to verify the connection.



Click **Next**. The Tier Naming Decision window opens.

## To map business applications, tiers, and nodes to your application environment

1. Read about how AppDynamics uses business applications, tiers, and nodes to organize application performance monitoring. In summary:

- A business application is a set of modules and distributed services that together provide business functionality.
- A node is the basic unit of processing that AppDynamics monitors.
- A tier represents a module in an application environment, such as an eCommerce website or Inventory application.

See Logical Model.

2. Decide how to identify and name the tiers. Either AppDynamics will automatically configure tier names, or you can manually configure them.

Use these guidelines for deciding whether to use automatic or manual naming:

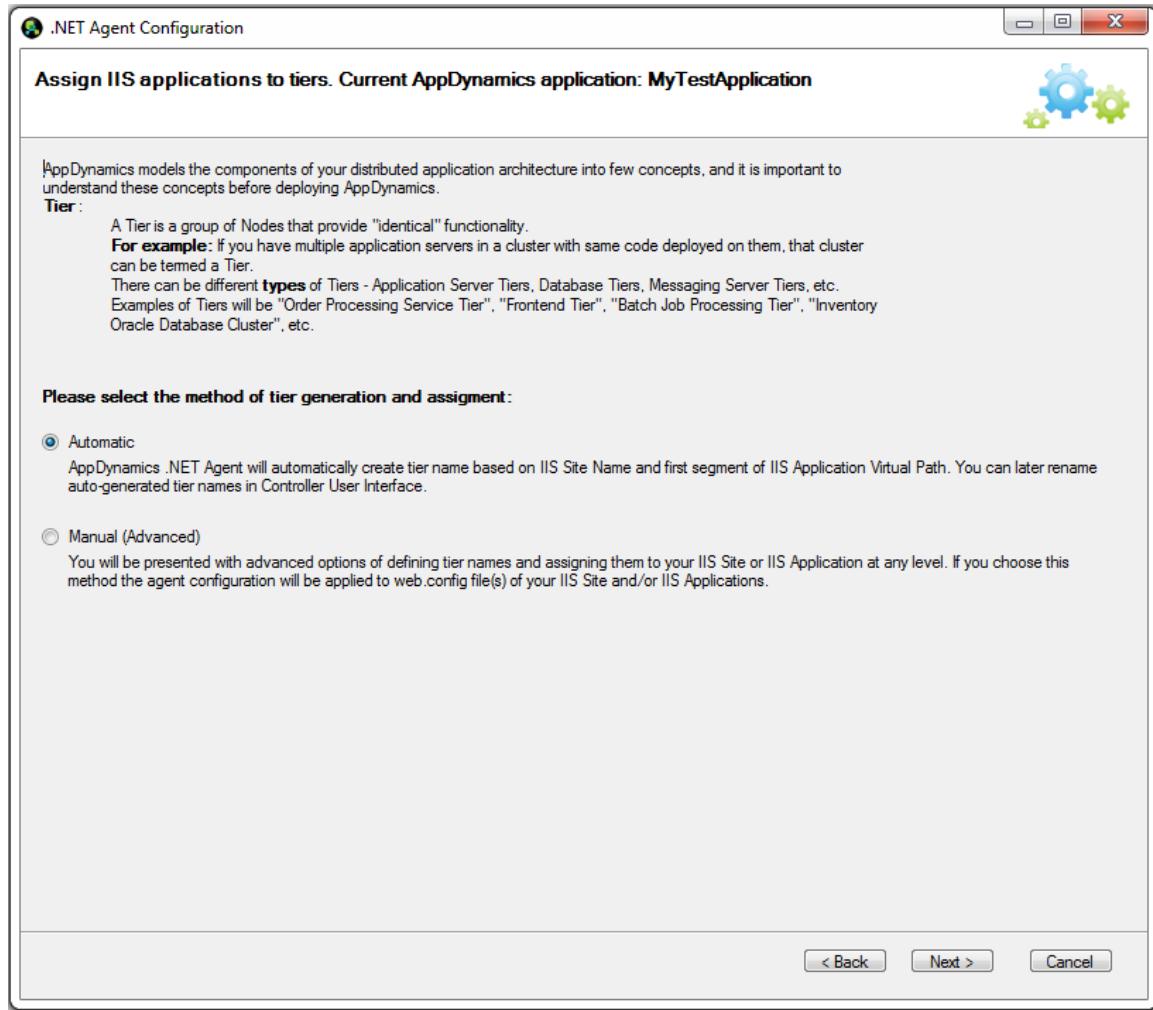
- When all IIS applications on a machine will be instrumented by AppDynamics, choose **automatic**.
- AppDynamics names tiers using this pattern:

IIS\_site\_name-IIS\_application\_name

- When only some IIS applications on a machine will be instrumented by AppDynamics, choose **manual**.
- In manual naming, you can name nodes as you like and AppDynamics will update the web.config file.

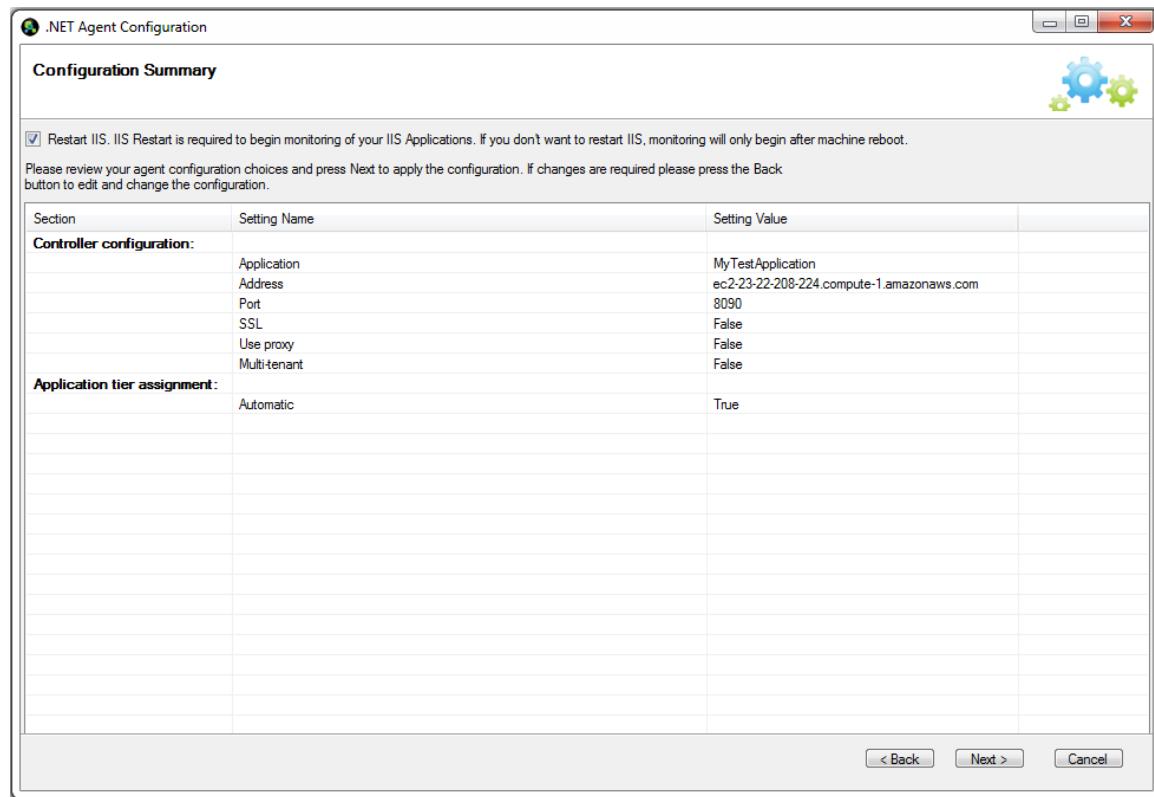
## To let AppDynamics automatically name the tiers

1. In the Tier Naming Decision window click **Automatic**.

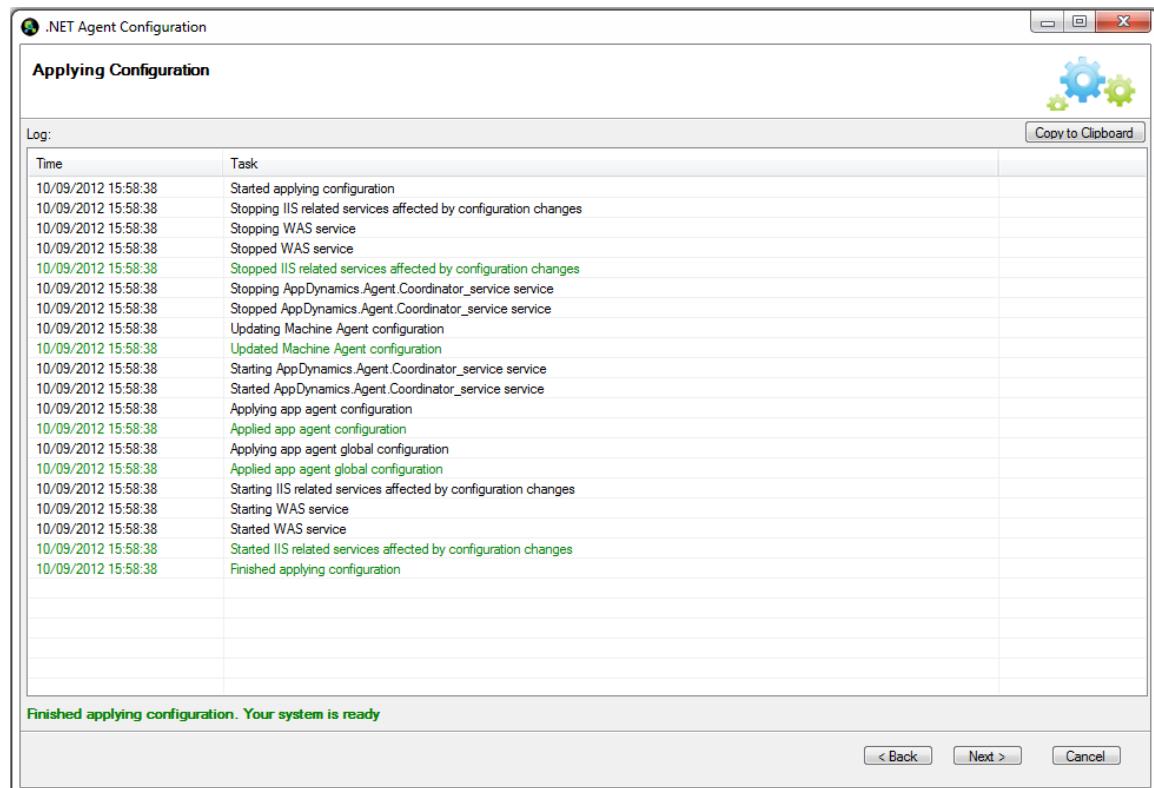


The Configuration Utility summarizes the configuration settings.

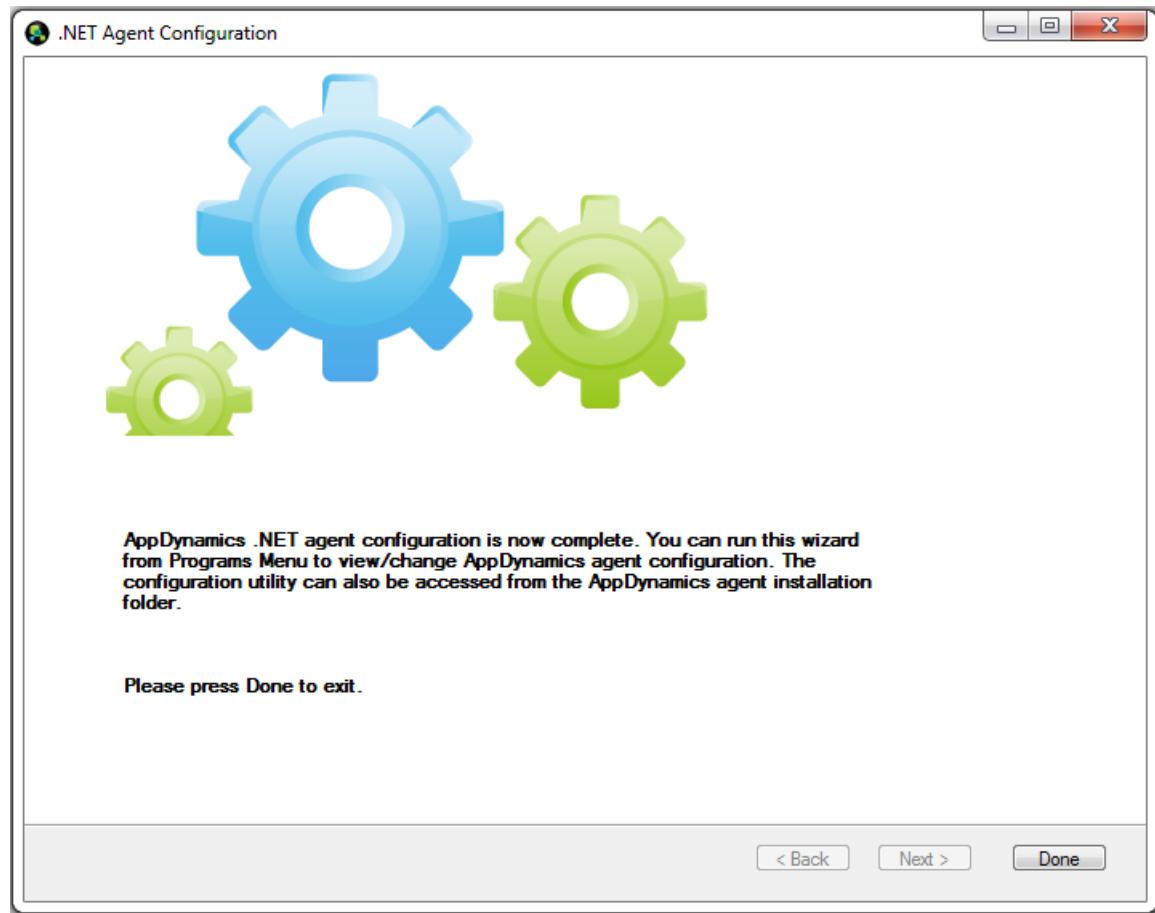
**⚠** By default when you click **Next** the Configuration Utility will restart IIS. If you do not want to apply the configuration right away, uncheck the box. The Configuration Utility will save the information and apply it the next time you restart IIS.



2. If you proceed and click **Next**, the Configuration Utility logs its activities, including stopping and restarting IIS, and reports any problems. Review the summary for any issues in red font. Green font indicates the more interesting logged events. The summary shows any Warnings (W) or Errors (E). If you have errors, contact [AppDynamics Support](#).



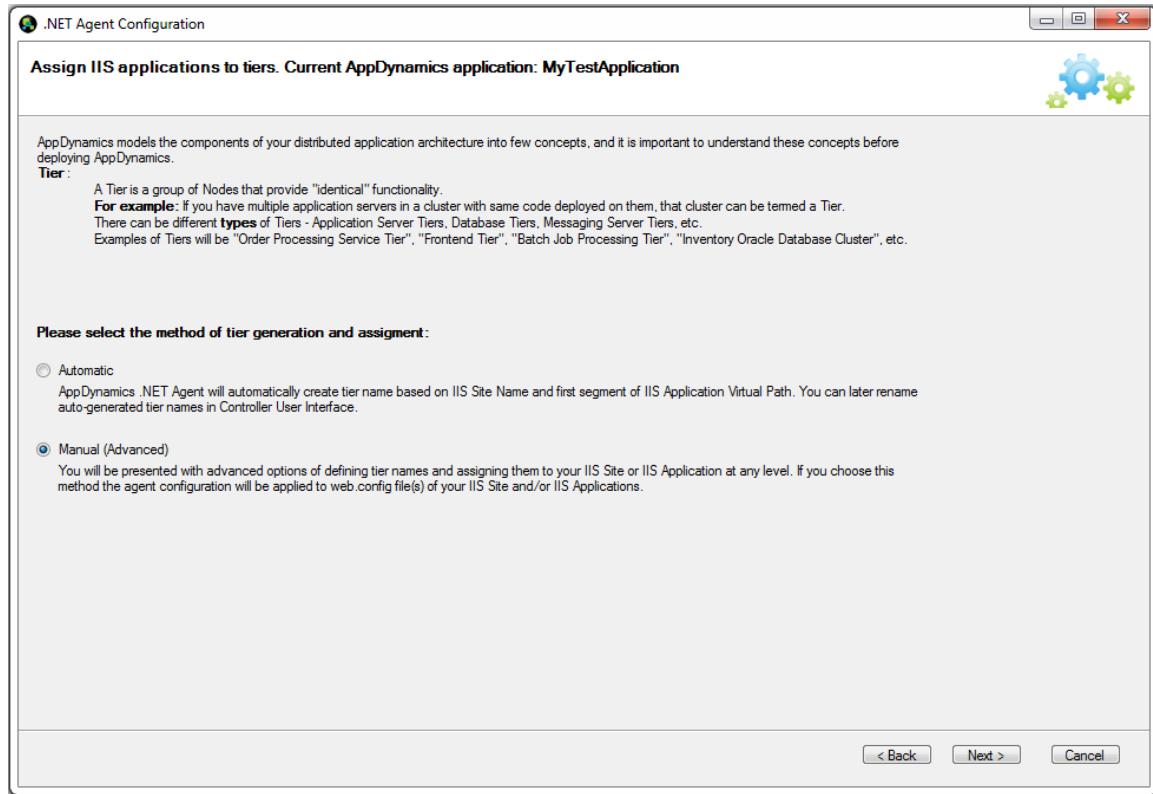
3. When there are no errors, click **Next**.



4. Click **Done** to close the Configuration Utility.

## To manually name the tiers

1. In the Tier Naming Decision window click **Manual**.

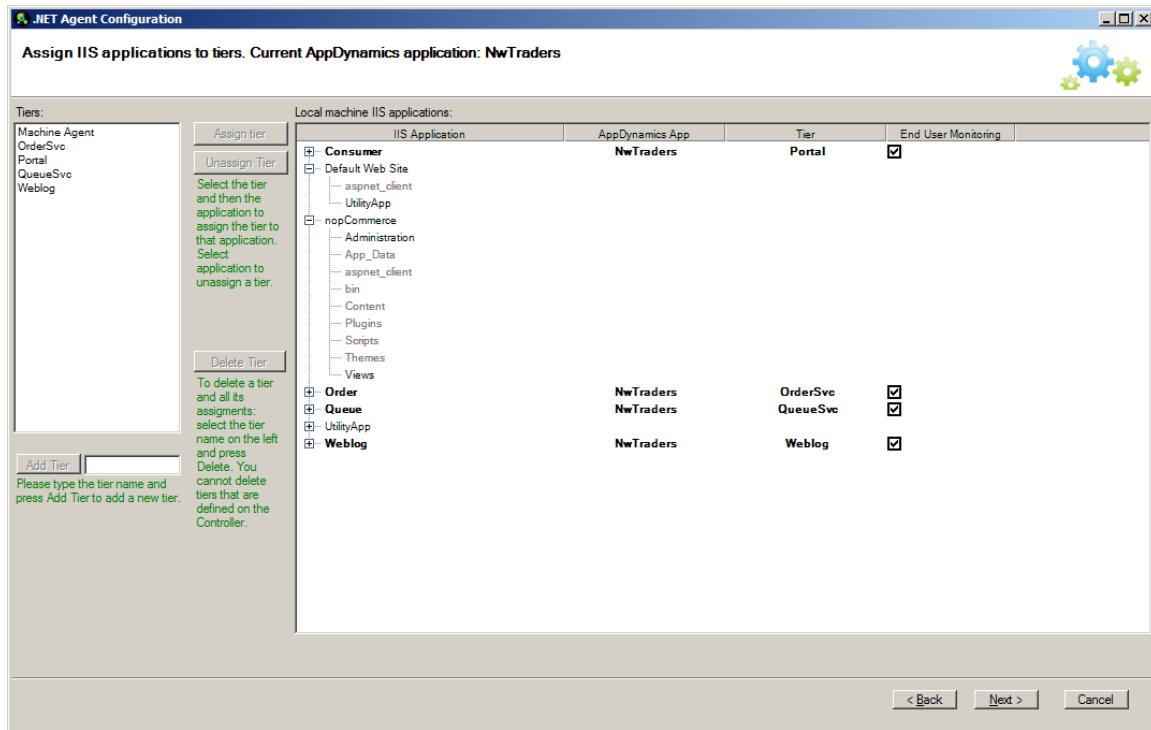


AppDynamics retrieves existing business application information from the Controller and displays it in the left column, and connection status in the right column.

If no business applications are already defined the utility displays an empty list.

2. Click **New Application** to define a new business application. Be careful about spellings and capitalization and note down the exact name.

**Note:** Do not use ampersands in the business application name; they are not supported at this time.



3. Click **Next**.

4. Assign IIS Applications to AppDynamics tiers.

Select a tier on the right and click a business application on the left. The assigned tier will be highlighted in boldface.

**i** Note: For large IIS installations, use the Max IIS tree depth pulldown to display all the projects. A large tree depth may take some time to view.

To add tiers that are not already configured, enter a name and click **Add Tier**.

5. When you are done click **Next**. AppDynamics displays a configuration summary.

6. Review the configuration. If you need to make changes click **Back**.

**!** By default when you click **Next** the Configuration Utility will restart IIS. If you do not want to apply the configuration right away, click \*Cancel\*. The Configuration Utility will save the information and apply it the next time you restart IIS.

**.NET Agent Configuration**

### Configuration Summary



**Note:** Agent configuration requires the installation to stop the Internet Information Services (IIS). Once the IIS services are shutdown all Web applications and services running under IIS will no longer be available to users.

Please review your agent configuration choices and press Next to apply the configuration. If changes are required please press the Back button to edit and change the configuration.

Section	Setting Name	Setting Value
<b>Controller configuration:</b>	Application	MyTestApplication
	Address	ec2-23-22-208-224.compute-1.amazonaws.com
	Port	8090
	SSL	False
	Use proxy	False
	Multi-tenant	False
<b>Application tier assignment:</b>	/ADServiceHost.Client	Cleaning configuration
	/BugBash.DynamicCode	Cleaning configuration
	/BugBash.MongoDB	Cleaning configuration
	/CustomerSuccess.SiteA	SiteA, EUM disabled
	/CustomerSuccess.SiteB	SiteB, EUM disabled
	/EUMTests.RegularWeb	Cleaning configuration
	/MorningStar.MatchEntrySite1	Cleaning configuration
	/MorningStar.MatchEntrySite2	Cleaning configuration
	/MorningstarWeb	Cleaning configuration
	/MSOtherWebsite	Cleaning configuration
	/MVC3EUMTest.Web	Cleaning configuration
	/POC.NavmanWireless.NWWeb	Cleaning configuration
	/POC.OpenTable.NetworkIO	Cleaning configuration
	/POC.Proflowers.Web	Cleaning configuration
	/POC.Proflowers.Webservice	Cleaning configuration

[< Back](#) [Next >](#) [Cancel](#)

7. If you proceed and click **Next**, the Configuration Utility logs its activities, including stopping and restarting IIS, and reports any problems. Review the summary for any issues in red font. Green font indicates the more interesting logged events. The summary shows any Warnings (W) or Errors (E). If you have errors, contact AppDynamics Support.

**.NET Agent Configuration**

**Applying Configuration**

Log:

Time	Task
10/09/2012 15:52:51	Started applying configuration
10/09/2012 15:52:51	Stopping IIS related services affected by configuration changes
10/09/2012 15:52:51	Stopping WAS service
10/09/2012 15:52:51	Stopped WAS service
10/09/2012 15:52:51	Stopped IIS related services affected by configuration changes
10/09/2012 15:52:51	Stopping AppDynamics.Agent.Coordinator_service service
10/09/2012 15:52:51	Stopped AppDynamics.Agent.Coordinator_service service
10/09/2012 15:52:51	Updating Machine Agent configuration
10/09/2012 15:52:51	Updated Machine Agent configuration
10/09/2012 15:52:51	Starting AppDynamics.Agent.Coordinator_service service
10/09/2012 15:52:51	Started AppDynamics.Agent.Coordinator_service service
10/09/2012 15:52:51	Applying app agent configuration
10/09/2012 15:52:51	Applying app agent configuration to application at: c:\users\petto\documents\visual studio 2010\Projects\CustomerSuccess\CustomerSuccess.SiteA\web.config
10/09/2012 15:52:51	Applied app agent configuration to application at: c:\users\petto\documents\visual studio 2010\Projects\CustomerSuccess\CustomerSuccess.SiteA\web.config
10/09/2012 15:52:51	Applying app agent configuration to application at: c:\users\petto\documents\visual studio 2010\Projects\CustomerSuccess\CustomerSuccess.SiteB\web.config
10/09/2012 15:52:51	Applied app agent configuration to application at: c:\users\petto\documents\visual studio 2010\Projects\CustomerSuccess\CustomerSuccess.SiteB\web.config
10/09/2012 15:52:51	Applied app agent configuration
10/09/2012 15:52:51	Applying app agent global configuration
10/09/2012 15:52:51	Applied app agent global configuration
10/09/2012 15:52:51	Starting IIS related services affected by configuration changes
10/09/2012 15:52:51	Starting WAS service
10/09/2012 15:52:51	Started WAS service
10/09/2012 15:52:51	Started IIS related services affected by configuration changes
10/09/2012 15:52:51	Finished applying configuration

Copy to Clipboard

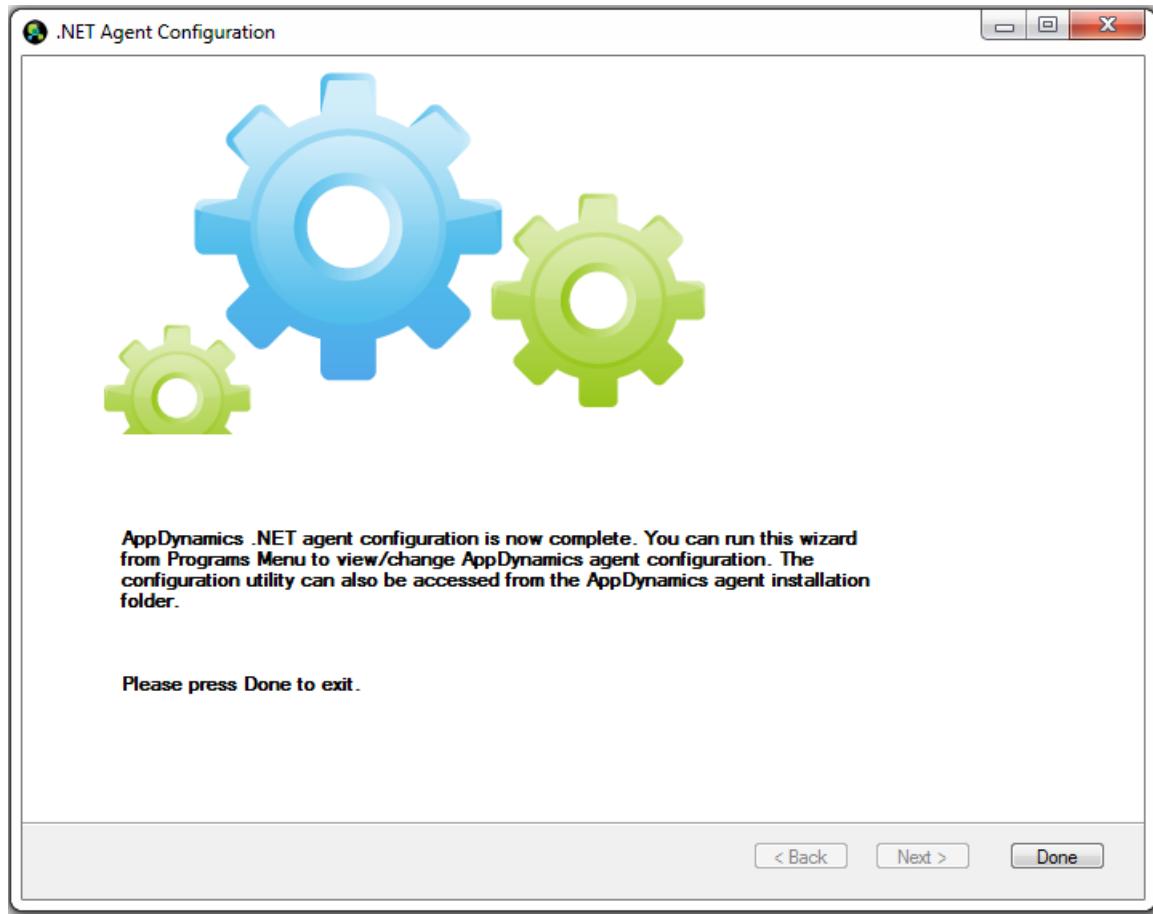
Finished applying configuration. Your system is ready

< Back Next > Cancel

8. Review the configuration log summary.

As it applies the configuration, AppDynamics generates a log of the configuration activities and displays a summary. Review the summary for any issues in red font. Green font indicates the more interesting logged events. The summary shows any Warnings (W) or Errors (E). If you have errors, contact AppDynamics Support.

9. Click **Next**. The wizard completes.



For troubleshooting information see [Troubleshoot App Agent for .NET Installation and Configuration](#).

## Using a Configuration File from the Command Line

You can set up a .NET Agent configuration file and run it from the command line. This is useful when you have multiple agents to configure.

### To create a configuration file

1. From a command line, start the configuration utility:

```
AppDynamics.Agent.Winston.exe -s <SetupConfigurationFilePath>
```

For example:

```
c:\Program Files\AppDynamics\AppDynamics .NET  
Agent\AppDynamics.Agent.Winston.exe -s  
"c:\temp\configurationSavedSetupConfiguration.xml"
```

The utility starts.

2. Configure the agent as described in the previous sections. The configuration is applied. In addition, when the configuration completes, AppDynamics creates a setup file.

You use this setup file as an argument to the command line utility.

## To run the configuration utility from the command line

1. Start the .NET Configuration Utility from the command line. Change the file path and setup file path as needed.

```
AppDynamics.Agent.Winston.exe -c <SetupConfigurationFilePath>
```

For example:

```
c:\Program Files\AppDynamics\appdynamics .NET  
Agent\appdynamics.agent.winston.exe -c  
"c:\temp\configurationSavedSetupConfiguration.xml"
```

The utility runs in command line mode; the user interface does not launch.

When it is done it exits the process with 0 for success or any other number for failure.

2. Review the Winston.txt log file in the default logs directory for details.

## Learn More

- [Install the App Agent for .NET](#)
- [Naming Conventions for .NET Nodes](#)

# How to Configure the .NET Agent Manually

*This topic was written by guest author Robert Petty of AppDynamics Customer Success. Thanks Robert!*

## Introduction

For instrumentation to occur you must define the configuration for the AppDynamics .NET Agent to properly be hooked inside the host (w3wp) process. In the case of IIS applications, this occurs via adding the appropriate agent configuration to the web.config. This configuration can reside higher in the hierarchical chain, but this how-to will show how to directly manipulate the IIS application at the same level in which it resides (the local web.config). This article assumes AppDynamics .NET Agent has already been successfully installed.

## Web.config Explanation

Regardless of how the original web.config is created there are two areas that are required for agent instrumentation. The two sections required are the section definition (which is a child of the <configSections> element. If it is not present you must add this parent element. Here it is in its entirety:

```
<configSections>  
  <section name="AppDynamicsAgent"  
    type="AppDynamics.Agent.Config.Section, AppDynamics.Agent,  
    Version=1.0.0, Culture=neutral, PublicKeyToken=3f604d9e4f8e4985"  
    allowLocation="true"    allowDefinition="Everywhere"    />  
</configSections>
```

The <section> element will alert the ASP.NET process, as it parses the web.config structure, from machine.config to local web.config,

which section to use for the base .NET AppDynamics Agent configuration. This is defined in the other relevant element, which resides at the same level as <configSections>, called <AppDynamicsAgent>:

```
<AppDynamicsAgent>
<controller-info>
  <controller-host>localhost</controller-host>
  <controller-port>8090</controller-port>
  <controller-ssl-enabled>false</controller-ssl-enabled>

  <!-- If the controller is running in multi-tenant mode, specify the account name
       and access key for this agent to authenticate with the controller.
       If the controller is running in single-tenant mode, there is no need to
       specify these values. -->
  <account-name></account-name>
  <account-access-key></account-access-key>

  <!-- For Auto Agent Registration specify the application name, tier name,
       and optionally, node name. If the application and/or tier does not
       exist already it will be created. -->
  <application-name>Test .NET Application</application-name>
  <tier-name>Portal</tier-name>

  <!-- Recommended to leave it empty if your application tier node has more than one instance,
       as in case of IIS applications running in web garden (app pool) or in the case when nodes
       of the same tier run on different machines.
       If unsure - leave it empty, It will be auto-generated-->
  <node-name></node-name>
  <proxy disable="true">
    <host></host>
    <port></port>
  </proxy>
</controller-info>
<app-agent-configuration>
  <configuration-properties>
    <property name='agent-overwrite' value='false'/>
  </configuration-properties>
  <agent-services>
    <agent-service name='BCIEngine' enabled='true'>
      <configuration-properties>
        </configuration-properties>
        <configuration>
          </configuration>
        </configuration>
      </agent-service>
    <agent-service name='CLRMetricsService' enabled='true'>
      <configuration-properties>
        <property name='update-interval-in-seconds' value='60'/>
      </configuration-properties>
      <configuration>
        <perf-counters>
          <!-- In this section you can add CLR related performance counters
              The list of possible .NET performance counters is documented here
              Performance Counters in the .NET Framework
              http://msdn.microsoft.com/en-us/library/w8f5kw2e%28v=VS.80%29.aspx -->
          <!-- Example: <counter cat=".NET CLR Exceptions" name="# of Exceptions Thrown"> -->
        </perf-counters>
      </configuration>
    </agent-service>
    <agent-service name='SnapshotService' enabled='true'>
      <service-dependencies>BCIEngine</service-dependencies>
      <configuration-properties>
        </configuration-properties>
      </agent-service>
    <agent-service name='TransactionMonitoringService' enabled='true'>
      <service-dependencies>BCIEngine,SnapshotService</service-dependencies>
      <configuration-properties>
        </configuration-properties>
      </agent-service>
    </agent-services>
  </app-agent-configuration>
  <interceptors>
```

```

<interceptor name="ADO.NET" disable="false"/>
<interceptor name="ASP.NET" disable="false"/>
<interceptor name="ASP.NET Web Services" disable="false"/>
<interceptor name="Directory Services Exit" disable="false"/>
<interceptor name="HttpWebRequest GetResponse exit" disable="false"/>
<interceptor name="Messaging exit receive" disable="false"/>
<interceptor name="Messaging exit send" disable="false"/>
<interceptor name="Remoting entry" disable="false"/>
<interceptor name="Remoting exit" disable="false"/>
<interceptor name="SOAP Exit" disable="false"/>
<interceptor name="WCF Entry" disable="false"/>
<interceptor name="WCF Exit" disable="false"/>
</interceptors>
<snapshots disable="false">
</snapshots>
</AppDynamicsAgent>

```

## Application Pool Configuration

The IIS application runs under the context of the account defined to run the application pool (impersonation being an exception inside the application itself). For the .NET Application Agent to write to the logs folder, it must run under an account that has access to the logs folder (generally C:\AppDynamics\Logs unless changed previously). AppDynamics best practices generally give the logs folder by default the following accounts permission when running the .NET Agent Configuration:

- ApplicationPool Identity
- Local Service
- Network Service
- System

Not all applications run under the ApplicationPoolIdentity so it is advised when manually configuring the .NET AppDynamics Agent to run the application pool under one of these accounts and ensure the same accounts have access to the logs folder.

## AppDynamics Configuration

The <AppDynamicsAgent> element is where you configure the controller communication for the application agent. The values here depend on whether the controller is on-premise, SaaS, single or multi-tenant, and if SSL is in use.  
In the web.config file, modify the following items in the <controller-info> element:

Item	Required	Default
<controller-host>	Yes	None
<controller-port>	Yes	<b>For On-premise Controller installations:</b> By default, port 8090 is used for HTTP and 8181 is used for HTTPS communication.  <b>For SaaS Controller service:</b> By default, port 80 is used for HTTP and 443 is used for HTTPS communication.

To configure the .NET Agent to use SSL see [App Agent for .NET Configuration Properties](#).

(For Multi-tenant mode or SaaS installations only) Configure the .NET Agent account information  
This step is required only when the AppDynamics Controller is configured in multi-tenant mode (the Controller is configured with multiple user accounts) or when you [Use a SaaS Controller](#). Skip this step if you are using single-tenant mode, which is the default for the on-premise installation.

In the web.config file, specify the properties for Account Name and Account Key. This information is provided in the Welcome email from AppDynamics Support Team.

Property	Required	Default
<account-name>	Required only if your Controller is configured for multi-tenant mode or your Controller is hosted.	None

<code>&lt;account-access-key&gt;</code>	Required only if your Controller is configured for multi-tenant mode or your Controller is hosted.	None
---	--	------

## Configure How The Agent Application And Tier Are Identified

In the web.config file, specify to which Application and Tier this .NET Agent belongs. For more information see [Mapping Application Services to the AppDynamics Model](#).

Property	Required	Default
<code>&lt;application-name&gt;</code>	Yes	None
<code>&lt;tier-name&gt;</code>	Yes	None

AppDynamics automatically identifies the node name as:

`<tier name><host name><app pool name><worker process index: 0,1,2...><clr index: 0,1,...><app domain index: 0,1,...>`

**Note:** Node names tend to have low values for the index and attempts are made to reuse the same node name after a recycle event (such as application pool recycle).

## Launching the Application Agent

If the machine agent has been configured and the relevant web.configs have been configured, it is now safe to launch the application. The steps are as follows:

1. Stop either the relevant application pools for the instrumented web application or stop IIS entirely.
2. Stop the AppDynamics.Agent.Coordinator\_service (described as AppDynamics.Agent.Coordinator).
3. Restart the application pool or IIS.
4. Place load on the application to start instrumentation and reporting to the defined AppDynamics controller application.

## Troubleshooting

To verify that instrumentation has occurred, visit the logs folder (generally c:\AppDynamics\Logs unless changed previously). This log folder contains the following if instrumentation has occurred:

- A folder called Data
- ByteCode.txt
- Possibly additional log files (AgentLog, BusinessTransactionLog, RESTHeartbeat, etc.)

If these are not seen, open AgentLog and look for lines containing w3wp and dllhost. If only dllhost is present and not w3wp, this means instrumentation has not occurred and generally will be due to the following issues:

- Incorrect permissions on the log folder (the application pool must have permission to write to the logs folder).
- A previous profiler has been installed. Even if removed, often software leaves behind residual hooks which can affect installs of other applications.

## Learn More

- Agent - Controller Compatibility Matrix
- Upgrade the App Agent for .NET
- Install the App Agent for .NET
- App Agent for .NET Configuration Properties

### Source

1. <http://msdn.microsoft.com/en-us/library/ms178685.aspx>
2. [Configure the .NET Machine Agent](#)

# App Agent for .NET Configuration Properties

- Properties to Configure the .NET App Server Agent
  - Configuration Scopes
- Required Agent Properties
  - Agent-Controller Communication Properties
  - Agent Identification Properties
  - Advanced Properties
    - SSL for the Controller
    - Agent Runtime Directory
    - Multi-tenant Mode
  - To use an existing Web.config file
- Learn More

This topic describes the .NET App Server Agent configuration properties.

## Properties to Configure the .NET App Server Agent

You can configure App Server Agent properties using either of the following options:

- The sample Web.config file provided at <agent\_install\_dir>\Samples\Configuration
- An existing Web.config file

## Configuration Scopes

There are different scopes to configuration settings:

- a global scope
- the scope of the application
- the root Web.config file

For details see [ASP.Net Configuration File Hierarchy](#).

Configuration Level	File Name	Notes
Root Web	Web.config	The Web.config file for the server is stored in the same directory as the Machine.config file and contains default values for most of the system.web configuration sections.
Web site	Web.config	The Web.config file for a specific Web site contains settings that apply to the Web site and inherit downward through all ASP.NET applications and subdirectories of the site.
ASP.NET application root directory	Web.config	The Web.config file for a specific ASP.NET application is located in the root directory of the application and contains settings that apply to the Web application and inherit downward through all of the subdirectories.
ASP.NET application sub directory	Web.config	The Web.config file for an application subdirectory contains settings that apply to this subdirectory and inherit downward through all of the subdirectories.
Client application directory	ApplicationName.config	The ApplicationName.config file contains settings for a Windows client application (not a Web application). The <ApplicationName> includes the .exe extension (app.exe.config).

## Required Agent Properties

Mandatory properties include:

- Agent-Controller Communication Properties configure how the Agent connects with the Controller.
- Agent Identification Properties configure how the Agent identifies itself.

Additionally, there are Advanced Properties, used for special cases.

## Agent-Controller Communication Properties

Modify the following properties in the Web.config file:

Description	Property	Required	Default
<b>Controller Host</b> <i>For an on-premise Controller installation: The value as configured for "Application Server Host Name".  For the SaaS Controller service, refer to the Welcome email.</i>	<controller-host>	Yes	None
<b>Controller Port</b> <i>For an on-premise Controller installation: Provide the value for "HTTP Listener Port". For SaaS Controller service, refer to the Welcome email.</i>	<controller-port>	Yes	<b>For on-premise:</b> Port 8090 is the default for HTTP and port 8181 is the default for HTTPS. <b>For SaaS Controller Service:</b> Port 80 is used for HTTP and port 443 is used for HTTPS.

For example:

```
<configurations>
<AppDynamicsAgent>
<controller-info>
    <controller-host>localhost</controller-host>
    <controller-port>8090</controller-port>
    ...
</controller-info>
</AppDynamicsAgent>
</configurations>
```

## Agent Identification Properties

Provide information about the business application and tier in the Web.config file:

Description	Property	Required	Default
Application Name <i>Name of the business application</i>	<application-name>	Yes	None
Tier Name <i>Name of the tier/cluster to which this Agent belongs</i>	<tier-name>	Yes	None

For example:

```
<configurations>
<AppDynamicsAgent>
<controller-info>
    .....
    <application-name>ACMEOnline</application-name>
    <tier-name>InventoryTier</tier-name>
    .....
</controller-info>
</AppDynamicsAgent>
</configurations>
```

## Advanced Properties

Advanced properties configure the Agent for special cases.

## SSL for the Controller

Use this property to enable or disable SSL for Agent-Controller communication.

Property	Default Value
<controller-ssl-enabled>	False (Allowed values: True, False)

## Agent Runtime Directory

Use this property to specify the runtime directory for all runtime files (logs, transaction configuration) for those nodes that use this Agent. If this property is specified, all Agent logs are written to <agent-runtime-dir>/logs/node-name and transaction configuration is written to <agent-runtime-dir>/conf/node-name.

Property	Default Value
<agent-runtime-dir>	<agent_install_dir>/nodes

## Multi-tenant Mode

Use these properties to specify the account name and account key in a multi-tenant setup.

**i** If you are using the SaaS Controller Service, these properties are mandatory. The <account-name> is provided to you in the Welcome email sent by the AppDynamics Support Team.

Property	Default Value
<account-name>	None
<account-access-key>	None

## To use an existing Web.config file

You can update an existing web.config file present in your system.

1. Go to the <configSections> element in your web.config file and add following section:

```
<configSections>
    <section name="AppDynamicsAgent"
        type="AppDynamics.Agent.Config.Section, AppDynamics.Agent,
        Version=1.0.0.0, Culture=neutral, PublicKeyToken=3f604d9e4f8e4985"
        allowLocation="true"          allowDefinition="Everywhere"      />
</configSections>
```

2. Add the following <AppDynamicsAgent> section:

```
<AppDynamicsAgent>
    <controller-info>
        <controller-host>host</controller-host>
        <controller-port>8090</controller-port>
        <controller-ssl-enabled>false</controller-ssl-enabled>

        <!-- If the controller is running in multi-tenant mode, specify the account name
            and access key for this agent to authenticate with the controller. -->
    </controller-info>
</AppDynamicsAgent>
```

```

If the controller is running in single-tenant mode, there is no need to
specify these values. -->
<account-name></account-name>
<account-access-key></account-access-key>

<!-- For Auto Agent Registration specify the application name, tier name,
and optionally, node name. If the application and/or tier does not
exist already it will be created. -->
<application-name>Test .NET Application</application-name>
<tier-name>Portal</tier-name>

<!-- Recommended to leave it empty if your application tier node has more than one
instance,
as in case of IIS applications running in web garden (app pool) or in the case
when nodes
of the same tier run on different machines.
If unsure - leave it empty, It will be auto-generated-->
<node-name></node-name>
<proxy disable="true">
<host></host>
<port></port>
</proxy>
</controller-info>
<app-agent-configuration>
<configuration-properties>
<property name='agent-overwrite' value='false' />
</configuration-properties>
<agent-services>
<agent-service name='BCIEngine' enabled='true'>
<configuration>
</configuration>
</agent-service>
<agent-service name='CLRMetricsService' enabled='true'>
<configuration>
<property name='update-interval-in-seconds' value='60' />
</configuration>
<perf-counters>
<!-- In this section you can add CLR related performance counters
The list of possible .NET performance counters is documented here
Performance Counters in the .NET Framework
http://msdn.microsoft.com/en-us/library/w8f5kw2e%28v=VS.80%29.aspx -->
<!-- Example: <counter cat=".NET CLR Exceptions" name="# of Exceps Thrown" /> -->
</perf-counters>
</configuration>
</agent-service>
<agent-service name='SnapshotService' enabled='true'>
<service-dependencies>BCIEngine</service-dependencies>
<configuration>
</configuration>
</agent-service>
<agent-service name='TransactionMonitoringService' enabled='true'>
<service-dependencies>BCIEngine,SnapshotService</service-dependencies>
<configuration>
</configuration>
</agent-service>
</agent-services>
</app-agent-configuration>
<interceptors>
<interceptor name="ADO.NET" disable="false"/>
<interceptor name="ASP.NET" disable="false"/>
<interceptor name="ASP.NET Web Services" disable="false"/>
<interceptor name="Directory Services Exit" disable="false"/>

```

```
<interceptor name="HttpWebRequest GetResponse exit" disable="false"/>
<interceptor name="Messaging exit receive" disable="false"/>
<interceptor name="Messaging exit send" disable="false"/>
<interceptor name="Remoting entry" disable="false"/>
<interceptor name="Remoting exit" disable="false"/>
<interceptor name="SOAP Exit" disable="false"/>
<interceptor name="WCF Entry" disable="false"/>
<interceptor name="WCF Exit" disable="false"/>
</interceptors>
<snapshots disable="false">
</snapshots>
```

```
</AppDynamicsAgent>
```

## Learn More

- [Install the App Agent for .NET](#)
- [Troubleshoot App Agent for .NET Installation and Configuration](#)

# Configure Authentication, User Permissions and Integrations

You can configure AppDynamics Controller user authentication and permissions in the Administration setup menu.

## Information on Configuring Authentication, Permissions, and Integrations

See [Configure Authentication Provider](#) for information on enabling an authentication provider (external or AppDynamics).

See [Configure Authentication Using SAML](#) if you are using SAML to authenticate users.

See [Configure Authentication Using LDAP](#) if you are using LDAP to authenticate users.

See [Configure Users](#) if you are using AppDynamics to authenticate users.

See [Configure Groups](#) to create groups to assign roles to multiple users whom you want to have identical privileges.

See [Configure Roles](#) to configure user permissions, for example which applications or custom dashboards a user can view, delete, or edit and which types of configurations a user can perform.

AppDynamics provides a set of predefined roles that you cannot modify. To create custom roles that assign permissions according to your organization's needs, see [Configure Custom Roles](#).

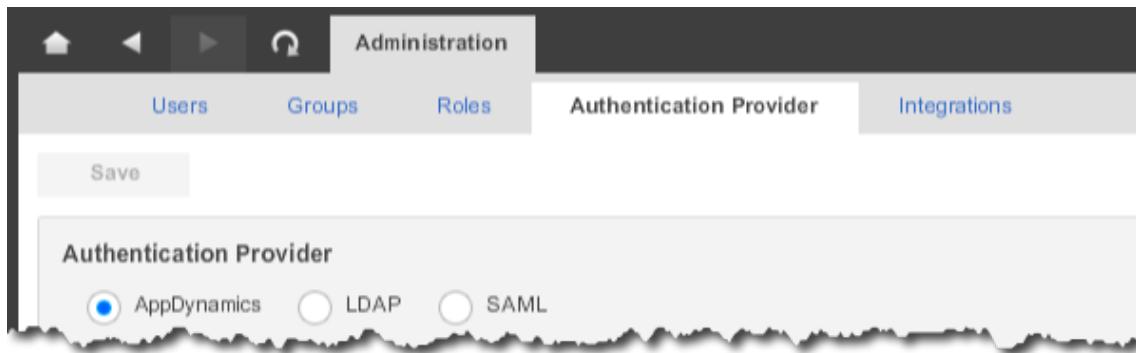
See [Configure Integrations](#) to enable and configure integrations with external products.

## To access the configuration screen

1. Click the Setup menu in the upper right section of the screen.

2. From the drop-down menu, click **Administration**.

The Administration screen opens with tabs for configuring the various settings.



## Configure Authentication Provider

- [To Enable an Authentication Provider](#)
- [Learn More](#)

AppDynamics Controller users can be authenticated by the AppDynamics Controller or by a third-party authentication system that is integrated with the Controller, such as LDAP or SAML.

The most efficient way to add and configure a large number of users is to use a third-party authentication provider.

See [To access the configuration screen](#) if you do not know how to access the administration configuration screens.

### To Enable an Authentication Provider

1. In the Administration screen, click the **Authentication Provider** tab.
2. Select radio button for the authentication provider to use.

If you select AppDynamics, the AppDynamics Controller is the authentication provider. In this case, users log in with their AppDynamics credentials. See [Configure Users](#) for information about creating AppDynamics users. You can also select one of the third-party authentication providers: LDAP or SAML.

### Learn More

- [Configure Authentication Using LDAP](#)
- [Configure Authentication Using SAML](#)
- [Configure Roles](#)
- [Configure Users](#)
- [Configure Groups](#)

## Configure Authentication Using SAML

- [Prerequisites](#)
- [Configuring SAML Settings](#)
  - [To Access Security Settings](#)
  - [To Enable a Security Provider](#)
  - [To configure AppDynamics SAML Settings](#).
- [Learn More](#)

This topic describes how to configure an AppDynamics account for single sign-on to an AppDynamics Controller using SAML.

The AppDynamics Controller uses this configuration to redirect authentication requests to an identity provider.

The AppDynamics SAML integration conforms to the Security Assertion Markup Language 2.0 (SAML 2.0) specification, so any SAML 2.0-compliant identity provider can be integrated with AppDynamics.

The specific configuration tasks depend on the SAML identity provider's implementation. See [SAML Configuration for OneLogin](#) for an example of AppDynamics' integration with One Login.

## Prerequisites

To perform SAML configuration you must have:

### 1. An account with a supported identity provider

You need your SAML Login URL and your x.509 certificate supplied by your identity provider.

If OneLogin is your identity provider, use the widget described in [SAML Configuration for OneLogin](#) to configure OneLogin for AppDynamics. For other SAML identity providers, you need to provide the following SAML authentication assertion custom attributes to be sent by the identity provider:

- accountName - required only if the controller is multi-tenant
- emailAddress - the user's email address.

AppDynamics assumes that the SAML authentication assertion NameID will be set to the user name of the user logging into the controller.

### 2. An account on an AppDynamics SaaS or on-premise Controller

In the case of an on-premise controller, if the controller is within a closed network and the identity provider is a service external to that network, the identity provider must be able to connect to the SAML Consumer URL and the controller must be able to connect to the SAML Login URL. For examples of these URLs, see [To configure AppDynamics SAML Settings for OneLogin](#) and [To configure OneLogin Settings for AppDynamics](#).

### 3. Account Administrator privileges on the AppDynamics Controller

Before configuring the SAML settings in the AppDynamics Controller, verify that the Account Administrator user who will log into the Controller and set up the SAML authentication also exists as a user in the identity provider. After configuring the SAML settings in AppDynamics and logging out, this user will be forced to log in again as that Account Administrator this time using the identity provider for authentication.

## Configuring SAML Settings

Follow the configuration steps specific to your identity provider to enable the integration.

### To Access Security Settings

1. Click the Setup menu in the upper right section of the UI.
2. From the drop-down menu, select **Administration**.

The Administration screen opens with tabs for configuring the various settings.

### To Enable a Security Provider

1. Click the **Authentication Provider** tab.
2. Select the **SAML** radio button for the authentication provider to use.

### To configure AppDynamics SAML Settings.

See [To configure AppDynamics SAML Settings for OneLogin](#) for a sample configuration with an actual identity provider.

The screenshot shows the 'Security Configuration' interface with the 'Authentication Provider' tab selected. At the top, there are tabs for 'Users', 'Groups', 'Roles', and 'Authentication Provider'. Below the tabs, there is a 'Save' button. The 'Authentication Provider' section has three radio button options: 'AppDynamics' (disabled), 'LDAP' (disabled), and 'SAML' (selected). The 'SAML' section contains fields for 'Login URL' and 'Logout URL', both of which are currently empty. Below these fields is a 'Certificate' area containing the text '-----BEGIN CERTIFICATE-----' and '-----END CERTIFICATE-----'. At the bottom of the SAML section is a table titled 'Default Roles' with columns for 'Member', 'Name', and 'Description'. The table lists five roles: 'Account Owner (Default)', 'Administrator (Default)', 'Custom Dashboard Viewer (Default)', 'Read Only User (Default)', and 'Workflow Executor (Default)'. Each role has a checkbox next to it under the 'Member' column.

In the SAML Configuration page:

1. In the Login URL field, enter the SAML Login URL. The SAML Login URL is the URL to the SSO service at the identity provider. The identity provider provides this URL to the Controller.
2. In the Logout URL field, enter the URL to which the browser should redirect when the user logs out. This is useful for redirecting the user back to the identity provider instead of to the AppDynamics login screen. This field is optional.
3. In the Certificate field, paste the x.509 certificate from your identity provider configuration between the BEGIN CERTIFICATE and END CERTIFICATE delimiters. Do not copy the BEGIN CERTIFICATE and END CERTIFICATE from certificate field.
4. In the Default Roles section, select the roles to grant to new users of the SAML-enabled controller by checking the Member check box for the role. You must grant at least one default role, and you can select multiple roles. See [Configure Roles](#) for information about roles and permissions.

The roles that you assign here will be granted to new users when they first log in to the SAML-enabled controller if those users have not been previously created directly in the Controller. Users created prior to SAML enablement or directly within the controller prior to the user's initial login retain their original roles.

Typically SAML users get the default roles assigned in this configuration. In exceptional cases an account owner may want to grant individual users different roles. See [To Assign A Role to a User](#).

5. Click **Save**.

## Learn More

- [SAML Configuration for OneLogin](#)
- [Disable SAML Authentication for an Account](#)
- [Configure Users](#)

## Disable SAML Authentication for an Account

- To disable SAML Authentication

You can disable SAML authentication for an account from the AppDynamics administration console.

### To disable SAML Authentication

1. Log into the administration console.  
See [Access the Administration Console](#).
2. In the left navigation panel click **Accounts**.
3. From the accounts list select the account for which you want to disable SAML authentication.
4. Click the Edit icon.  
The account settings for the account are displayed. If SAML is enabled, the setting appears as follows.

Using SAML Authentication	Yes	<a href="#">Disable</a>
---------------------------	-----	-------------------------

5. Click **Disable**.

## SAML Configuration for OneLogin

- To configure AppDynamics SAML Settings for OneLogin
- To configure OneLogin Settings for AppDynamics
- [Learn More](#)

This topic describes a sample SAML integration with the identity provider, OneLogin. See [Configure Authentication Using SAML](#) for general information about SAML integration.

### To configure AppDynamics SAML Settings for OneLogin

1. Access the security settings configuration screen. See [To Access Security Settings](#).
2. Select **SAML** as the provider. See [To Enable a Security Provider](#).  
The SAML Single Sign On Configuration page opens.

Member	Name	Description
<input type="checkbox"/>	Account Owner (Default)	Can access and modify global configuration and view/modify applications and dashboards
<input type="checkbox"/>	Administrator (Default)	Can view/modify applications and dashboards
<input type="checkbox"/>	Custom Dashboard Viewer (Default)	Can view dashboards only
<input type="checkbox"/>	Read Only User (Default)	Can view applications and dashboards but not modify their configuration
<input type="checkbox"/>	Workflow Executor (Default)	Can execute workflows only

3. In the Login URL field enter the SAML Login URL from your OneLogin configuration. The SAML Login URL is the URL to the SSO

service at the identity provider. The identity provider provides this URL to the Controller.

If you do not know your SAML Login URL, you can locate it in your OneLogin configuration:

- a. Log in to your OneLogin account.
- b. Click the **Apps** tab in the first set of tabs.

c. Click **edit** next to the application for which you want to view the Login URL.

Click the **Company Apps** tab in the second set of tabs if it is not already selected.

d. Click **Single Sign-on** in the third set of tabs.

The SAML Login URL is the HTTP SAML Endpoint in the Sign-on method section.

The screenshot shows the AppDynamics configuration page within the OneLogin interface. The top navigation bar includes tabs for Portal, Apps, People, Activity, Security, and Account. Below this, a secondary navigation bar has tabs for Company apps, Find apps, Tabs, and Custom connectors. The main content area is titled 'AppDynamics' and shows the following configuration details:

Sign-on method	SAML2.0
Issuer URL	<a href="https://app.onelogin.com/saml/metadata/46533">https://app.onelogin.com/saml/metadata/46533</a>
SAML Endpoints:	 HTTP: <a href="https://app.onelogin.com/trust/saml2/http-post/sso/46533">https://app.onelogin.com/trust/saml2/http-post/sso/46533</a> SOAP: <a href="https://app.onelogin.com/trust/saml2/soap/sso/46533">https://app.onelogin.com/trust/saml2/soap/sso/46533</a>
Credentials	<input checked="" type="radio"/> Configured by admin <input type="radio"/> Shared
Default values	Login AD user name Email Address Email

4. In the Logout URL field in the AppDynamics form, enter the URL to which the browser should redirect when the user logs out. This is useful for redirecting the user back to the identity provider instead of to the AppDynamics login screen. This field is optional. For example, the following logout URL redirects the user to the OneLogin application dashboard:

<https://app.onelogin.com/client/apps>

5. In the Certificate field in the AppDynamics form, paste the x.509 certificate from your OneLogin configuration between the BEGIN CERTIFICATE and END CERTIFICATE delimiters. Do not copy the BEGIN CERTIFICATE and END CERTIFICATE from the OneLogin x.509 certificate field.

To find your x.509 certificate in your OneLogin configuration:

- a. Log in to your OneLogin account.
- b. Click the **Security** tab in the first set of tabs.
- c. Click **SAML** in the second set of tabs.



The x.509 certificate is generated specifically for your OneLogin account and must be entered in any external application that you wish to use SAML with. Configuring SAML can be done in two steps.

1. Add an application that uses SAML, e.g. Google Apps or Salesforce
2. Configure the application with the x.509 certificate using the [Instructions here](#)

#### x.509 certificate

```
-----BEGIN CERTIFICATE-----
MIICMTCCAiWgAwIBAgIBATADBgEAMCcxCzAJBgNVBAYTAlVTMRMwEQYDVQQIDAoD
-----END CERTIFICATE-----
```

6. In the Default Roles section in the AppDynamics form, select the roles to grant to new users of the SAML-enabled controller by checking the Member check box for the role. You can select multiple roles in the list. See [Configure Roles](#) for information about roles and permissions.

The roles that you assign here will be granted to new users when they first log in to the SAML-enabled controller if those users have not been previously created directly in the Controller. Users created prior to SAML enablement retain their original roles.

You must grant at least one default role.

7. Click **Save**.

### To configure OneLogin Settings for AppDynamics

In your OneLogin account, configure the SAML Consumer URL with the host, port and optional account name values from the AppDynamics Controller. The Consumer URL is where the identify provider posts the SAML Authentication Assertion.

1. Log in to your OneLogin account.
2. Click the **Apps** tab in the first set of tabs.
3. Click **edit** next to the AppDynamics Connector.  
Click the **Company Apps** tab in the second set of tabs if it is not already selected.
4. In the third set of tabs click **Configuration**.

5. Enter the Consumer URL for the AppDynamics connector.  
It has the format:

`http[s]://<controller-host>:<controller-port>/controller/saml-auth`

The host and port for your Controller account are supplied by AppDynamics.

Application settings	Consumer URL
	<input type="text" value="https://test-huge4.saas.appdyn..."/>
	Account Name
	<input type="text"/>
	needed only for multi-tenant AppDynamics controllers
<input type="button" value="Cancel"/> <input type="button" value="Delete"/> <input type="button" value="Update"/>	

6. Provide the AppDynamics account name if your controller is configured in multi-tenant mode and if the user normally enters an account name on login. If your controller is configured in single-tenant mode or if the user does not supply an account name on login, you can leave the Account Name field blank.

See [Controller Tenant Mode](#) for information about controller tenant modes.

7. Save your settings.

## Learn More

- [Configure Authentication Using SAML](#)
- [Disable SAML Authentication for an Account](#)

# Configure Authentication Using LDAP

- [Prerequisites](#)
- [Configuring LDAP Settings](#)
  - [To Access the LDAP Configuration Screen](#)
  - [To Enable LDAP Authentication for AppDynamics](#)
  - [To Configure the Connection to the LDAP Server](#)
  - [To Configure the Query to Find Users](#)
  - [To Configure the Query to Find Groups](#)
- [Configuring the LDAP Synchronization Frequency](#)
  - [To Configure the LDAP Synchronization Frequency](#)
- [Configuring the Maximum Entries Returned from the LDAP Server](#)
  - [To Configure the Maximum LDAP Results Limit](#)
- [Using LDAP](#)
  - [To Enable LDAP as Your Authentication Provider](#)
  - [To Assign AppDynamics Permissions to an LDAP User](#)
  - [To Assign AppDynamics Permissions to an LDAP Group](#)
- [Learn More](#)

This topic describes how to configure AppDynamics account authentication using Lightweight Directory Access Protocol (LDAP).

The AppDynamics Controller uses this configuration to redirect authentication requests to an LDAP server.

## Prerequisites

To perform LDAP configuration you must have:

- An LDAP server  
There is a one-to-one correspondence between an AppDynamics account and an LDAP server.
- An account on an AppDynamics SaaS or on-premise Controller
- Account Administrator privileges on the AppDynamics Controller, as described in [Administrative Users](#).

## Configuring LDAP Settings

Configuring LDAP settings includes:

- Enabling or disabling LDAP authentication for AppDynamics
- Configuring the connection to the LDAP server
- Configuring the query LDAP uses to find users
- Configuring the query LDAP uses to find groups

### To Access the LDAP Configuration Screen

1. Click the Setup menu in the upper right section of the screen.
2. From the drop-down menu, click **Administration**.
3. Click **Authentication Provider**.

4. Select **LDAP**.

The LDAP configuration screen opens.

## To Enable LDAP Authentication for AppDynamics

Select LDAP in the security settings configuration screen. See [To Enable a Security Provider](#).

## To Configure the Connection to the LDAP Server

1. Access the LDAP configuration screen if you have not yet done so. See [To Access the LDAP Configuration Screen](#).

2. In the General section, you can optionally set the paging feature:

- Enable Paging: Check this option if the number of entries that could be pulled in exceeds the limits on the LDAP server.
- Page Size: Number of entries per round-trip from AppDynamics to LDAP; 500 is the default. A maximum size would depend on the LDAP server configuration.

3. In the Connection String section of the LDAP configuration screen, enter the information AppDynamics uses to connect to the LDAP server:

- Host: Address of the LDAP server. Required.
- Port: Port that the LDAP server listens on. Default is 636 for an SSL connection and 389 if not using SSL. Required.
- Use SSL: Enabled by default to use a secure connection to the LDAP server. Clear if not using SSL.
- Enable Referrals: Enabled by default to support LDAP referrals. For an overview of LDAP referrals see [Referrals in the LDAP](#).
- Maximum Referral Hops: The maximum number of referrals that AppDynamics will follow in a sequence of referrals. Default is 5. See [referral hop limit](#).
- Bind DN: Distinguished name of the AppDynamics administrative user to log into the LDAP Server. Required.
- Password: Password of the AppDynamics administrative user. Required.

4. Click **Test Connection** to ensure that the connection string works properly.

5. When the connection test is successful, click **Save**.

## To Configure the Query to Find Users

1. Access the LDAP configuration screen if you have not yet done so. See [To Access the LDAP Configuration Screen](#).

2. In the Users Query section of the LDAP configuration screen, enter the information to use to find LDAP users:

- Base DN: Location in the LDAP tree to begin recursively searching for users. Required.
- Filter: Optional LDAP search string that filters the items matched from the base DN. See [RFC2254](#) for information about LDAP search filters.
- Login Attribute: User's LDAP login. Default is "uid".
- Display Name Attribute: Optional user's display name.
- Group Membership Attribute: Optional user group membership. Recommended for faster retrieval.
- Email Attribute: Optional user email address.

▼ Users Query

Base DN	<input type="text" value="cn=users, dc=ad, dc=example, dc=com"/>
Filter	<input type="text"/>
Login Attribute	<input type="text" value="uid"/>
Display Name Attribute	<input type="text" value="displayName"/>
Group Membership Attribute	<input type="text" value="memberOf"/>
Email Attribute	<input type="text" value="mail"/>
<input type="button" value="Test Query"/>	

3. To test the configuration, click **Test Query**.

This launches a screen displaying the first few users returned by the query.

4. Click **Save**.

## To Configure the Query to Find Groups

LDAP Group configuration is optional.

1. Access the LDAP configuration screen if you have not yet done so. See [To Access the LDAP Configuration Screen](#).

2. In the Groups Query section of the LDAP configuration screen, enter the information to use to find LDAP groups:

- Base DN: Location in the LDAP tree to begin recursively searching for groups. Required.
- Enable Nested Groups: Option to include nested LDAP groups to a depth of 10.
- Filter: Optional LDAP search string that filters the items matched from the base DN. See [RFC2254](#) for information about LDAP search filters.
- Name Attribute: Name of the group. Default is "cn". Required.
- Description Attribute: Optional description of the group.
- User Membership Attribute: Identifies member of the groups. Optional.
- Referenced User Attribute: Optional Child attribute of the User Membership Attribute. Disabled if the parent is empty. Identifies property of the user that the user membership attribute contains.

▼ Groups Query

Base DN	<input type="text" value="ou=Groups,dc=corp,dc=appdynamics,dc=com"/>
Filter	<input type="text" value="objectClass=groupOfNames"/>
Name Attribute	<input type="text" value="cn"/>
Description Attribute	<input type="text" value="description"/>
User Membership Attribute	<input type="text" value="member"/>
Referenced User Attribute	<input type="text" value="dn"/>
<input type="button" value="Test Query"/>	

3. To test the configuration, click **Test Query**.

This launches a screen displaying the first few groups returned by the query.

4. When the query test is successful, click **Save**.

## Configuring the LDAP Synchronization Frequency

You can configure how often the Controller synchronizes with the LDAP server to capture new and removed users and groups. The default is 1 hour (3600000 milliseconds).

## To Configure the LDAP Synchronization Frequency

1. Stop the Controller application server:

On Linux:

```
./controller.sh stop-appserver
```

On Windows:

```
controller.bat stop-appserver
```

2. Open the <Controller-Installation-Directory>/appserver/domains/domain1/config/domain.xml file for edit.

3. In the <jvm-options> element, set the appdynamics.ldap.sync.frequency system property to the desired synchronization frequency in milliseconds.

For example, to synchronize every 15 minutes (900000 milliseconds) enter:

```
<jvm-options>-Dappdynamics.ldap.sync.frequency=900000</jvm-options>
```

4. Save the file.

5. Restart the Controller app server:

On Linux:

```
./controller.sh start-appserver
```

On Windows:

```
controller.bat start-appserver
```

## Configuring the Maximum Entries Returned from the LDAP Server

If you are importing a large number of LDAP users or groups into the Controller, it is possible to get a max\_results\_exceeded error. This occurs if your request exceeds the LDAP server's limit on the number of entries that an LDAP client can retrieve in a single operation.

If you do get the max\_results\_exceeded error, AppDynamics recommends that you first try to revise the filter in your user or group query to return fewer results. See [To Configure the Query to Find Users](#) and [To Configure the Query to Find Groups](#).

If you are unable to reduce the number of results returned so that you do not get the error, you can increment the appdynamics.controller.ldap.max.results system property to the number of entries that you expect to return, if that number is less than the LDAP server's hard limit on the number of entries that can be returned to the client. Work with your LDAP administrator to arrive at the right value.

If neither fine-tuning the user query filter nor setting the appdynamics.controller.ldap.max.results system property succeeds in stopping the max\_results\_exceeded error, call AppDynamics Support.

## To Configure the Maximum LDAP Results Limit

1. Stop the Controller application server:

On Linux:

```
./controller.sh stop-appserver
```

On Windows:

```
controller.bat stop-appserver
```

2. Open the <Controller-Installation-Directory>/appserver/domains/domain1/config/domain.xml file for edit.
3. In the <jvm-options> element, set the appdynamics.controller.ldap.max.results system property to a value somewhat greater than the number of entries that you expect to retrieve.  
For example, to set the maximum results returned to 800 enter:

```
<jvm-options>-Dappdynamics.controller.ldap.max.results=800</jvm-options>
```

4. Save the file.
5. Restart the Controller app server:

On Linux:

```
./controller.sh start-appserver
```

On Windows:

```
controller.bat start-appserver
```

## Using LDAP

You can assign permissions to LDAP groups or users.

### To Enable LDAP as Your Authentication Provider

In the **Authentication Provider** tab of the Security Configuration screen, select LDAP.  
See [Configure Authentication, User Permissions and Integrations](#).

### To Assign AppDynamics Permissions to an LDAP User

1. In the Security Configuration window, click the **Users** tab.  
If LDAP is enabled and correctly configured, AppDynamics fetches the usernames of the authenticated LDAP users.
2. Select the name of the user to whom you want to assign permissions.
3. In the Roles panel, check the roles that you want to assign to this user. You can assign multiple roles to a user.

The screenshot shows the 'User Configuration' screen. At the top, there are 'Save', 'Delete', and 'Create New User' buttons. Below that, a user profile for 'apatel' is shown with fields for 'Username' (apatel), 'Name' (Arpit Patel), and 'Email'. A 'Change Password' link is also present. The 'Groups' section shows one group, 'VPN\_Users'. The 'Roles' section lists five roles: Account Owner (Default), Administrator (Default), Custom Dashboard Viewer (Default), Read Only User (Default) (which is checked), and Workflow Executor (Default).

4. When the query test is successful, click **Save**.

## To Assign AppDynamics Permissions to an LDAP Group

LDAP Group configuration is optional. LDAP nested groups are not supported.

1. In the Security Configuration window, click the **Groups** tab.  
If LDAP is enabled and correctly configured, AppDynamics fetches the group names of the authenticated LDAP users.
2. Select the name of the group to which you want to assign permissions.
3. In the Roles panel, check the roles that you want to assign to this group. You can assign multiple roles to a group.
4. Click **Save**.

## Learn More

- Configure Roles
- Configure Groups
- Configure Users

## Configure Users

- To Create an AppDynamics User
- To Assign a User to a Group
- To Assign A Role to a User
- To Modify A User's Settings
- To Delete a User
- To Duplicate a User
- Learn More

This topic describes how to configure an AppDynamics Controller user if you are not using an external provider, such as LDAP or SAML, to authenticate AppDynamics Controller users.

## To Create an AppDynamics User

1. Click the Setup menu in the upper right section of the screen.
2. From the drop-down menu, click **Administration**.

3. Click the **Users** tab.
4. Click the Add icon.
5. Enter the information for the following fields:

- Username
- Name
- Email
- Password
- Repeat Password

6. Click **Save**.
7. Click **Create New User**.

## To Assign a User to a Group

1. In the left panel, select the user or users whom you want to assign to one or more groups. You can enter a string in the filter field to locate a specific user.
2. In the Groups list in the right panel, check the Member check boxes of the groups to which you want to assign the selected user or users. You can enter a string in the filter field to locate a specific group. You can click **Select All** to check all the groups and **Unselect All** to uncheck all the groups. A single user can be assigned to multiple groups.
3. Click **Save**.

A single user can belong to multiple groups.

You can also assign users to a group from the **Groups** tab. See [Configure Groups](#).

## To Assign A Role to a User

1. In the left panel, select the user or users to which you want to assign one or more roles. You can enter a string in the filter field to locate a specific user.
2. In the Roles list in the right panel, check the Member checkboxes of the roles that you want to assign to the selected user or users. You can enter a string in the filter field to locate a specific role. You can click **Select All** to check all the roles and **Unselect All** to uncheck all the roles. A single user can be assigned multiple roles.
3. Click **Save**.

You can also assign roles to a user from the **Roles** tab.

**Warning:** Do not remove yourself from all groups or from all roles. Also, if the only roles of which you are a member are custom roles, do not delete those custom roles or remove permissions from them. Doing these things can result in your being locked out of AppDynamics with no permissions at all. If this happens, contact AppDynamics Support.

## To Modify A User's Settings

1. In the User list in the left panel, select the user to modify.
2. Click the Edit icon.
3. To change the username, name and password, enter the new values in the fields in the right panel.
4. To change the password, click **Change Password** and then enter the new password values in the Password and Repeat Password fields in the right panel.
5. Click **Save**.

## To Delete a User

1. In the User list in the left panel, select the user to delete.
2. Click **Delete**.

## To Duplicate a User

1. In the User list in the left panel, select the user to duplicate.
2. Click the Duplicate icon.

In the Duplicate User screen overwrite the user's name with a new name and enter the new password.

The user created by duplication gets the groups and roles of the user from which he was cloned.

## Learn More

- [Configure Groups](#)
- [Configure Roles](#)

## Configure Groups

- [To Create an AppDynamics Group](#)
- [To Assign a User to a Group](#)
- [To Assign A Role to a Group](#)
- [To Modify A Group's Settings](#)
- [To Delete a Group](#)
- [Learn More](#)

This topic describes how to configure an AppDynamics Controller group. A group is a set of users that can be granted sets of permissions through assignment of roles. It is more efficient to assign roles to a group than to assign the same roles to each user individually.

If you are using LDAP to authenticate all AppDynamics Controller users you do not need to create AppDynamics groups.

### To Create an AppDynamics Group

1. Click the Setup menu in the upper right section of the screen.
2. From the drop-down menu, click **Administration**.
3. Click the **Groups** tab.
4. Click the Add icon.
5. Enter the information for the following fields:
  - Name
  - Description
6. Click **Save**.
7. Click **Create New Group**.

### To Assign a User to a Group

1. In the left panel, select the group to which you want to assign one or more users. You can enter a string in the filter field to locate a specific group.
2. In the Users list in the right panel, check the Member check boxes of the users whom you want to assign to the selected group or groups. You can enter a string in the filter field to locate a specific user. You can click **Select All** to check all the users and **Unselect All** to uncheck all the users.

A single user can belong to multiple groups.

### To Assign A Role to a Group

1. In the left panel, select the group or groups to which you want to assign roles. You can enter a string in the filter field to locate a specific group.

2. In the Roles list in the right panel, check the Member check boxes of the roles that you want to assign to the selected group or groups. You can click **Select All** to check all the roles and **Unselect All** to uncheck all the roles. You can enter a string in the filter field to locate a specific role.

A single group can be assigned multiple roles.

3. Click **Save**.

## To Modify A Group's Settings

1. In the groups list in the left panel, select the group to modify.
2. Enter the new values in the fields in the right panel.
3. Click **Save**.

## To Delete a Group

1. In the group list in the left panel, select group to delete.
2. Click **Delete**.

## Learn More

- [Configure Roles](#)
- [Configure Users](#)
- [Configure Authentication Using LDAP](#)
- [Configure Authentication Using SAML](#)

## Configure Roles

- [Predefined Roles](#)
- [Custom Roles](#)
- [Permission Inheritance for Default Permissions](#)
- [Account Level Permissions](#)
- [Application Level Permissions](#)
- [Custom Dashboard Permissions](#)
- [Learn More](#)

Roles define a set of privileges that AppDynamics Controller users have within the AppDynamics managed environment. This is also called "role-based access control", or "RBAC".

Roles provide an easy way to define and clone a set of permissions for a user or a group without having to configure every user's or every group's permissions individually.

You can assign a role to a user or group either in the **Users** or **Groups** tabs or in the **Users and Groups with this Role** subtab of the Roles configuration screen.

A user or group can have multiple roles.

## Predefined Roles

AppDynamics provides the following predefined roles:

- Account Owner: Can manage security settings (users, groups, roles) as well as view and modify applications and dashboards. This role is also known as the account administrator.
- Administrator: Can view and modify components that change state: applications, business transactions, dashboards, etc. Can view and edit all applications and all custom dashboards.
- Custom Dashboard Viewer: Can only view custom dashboards. Cannot do anything else.
- Read Only User: Can view but not edit all applications.
- Workflow Executor: Can execute workflows.

You can view the configurations for the predefined roles but you cannot change them. See [View Predefined Roles](#).

Although you cannot modify the predefined roles, you can add or remove users and groups from the predefined roles by checking or clearing the Member check box in the Roles panel. See [Configure Users](#) and [Configure Groups](#).

## Custom Roles

You can create custom roles if the predefined roles do not fit your organization's needs.

You can create a custom role from scratch or you can duplicate a predefined role, save it under another name, and then modify it as a custom role.

You can assign permissions to a custom role at the application/tier level, the custom dashboard level, and at the account level.

See [Configure Custom Roles](#).

## Permission Inheritance for Default Permissions

For application-level and custom dashboard permissions, AppDynamics provides a default set of permissions, named Default. The default permissions are inherited by new applications and new custom dashboards. The Default permissions are listed first in the **Application Permissions** subtab of the **Roles** tab.

Tier-level permissions are inherited from the containing application.

## Account Level Permissions

Permissions that can be granted at the account level include:

- Administer: Users with this permission can edit users, groups, roles, and the authentication provider.
- Configure Email/SMS: Users with this permission can edit email and SMS settings used by AppDynamics to send alerts. See [Configure the SMTP Server](#) and [Notification Actions](#).
- Execute Workflows: Users with this permission can execute workflows. See [Workflow Automation](#).

## Application Level Permissions

Permissions that can be granted at the application level and tier levels include the following. Follow the links if you want to see precisely the tasks that the configuration permissions allow the user to perform.

- Configure Business Transaction Detection
- Configure Backend Detection
- Configure Error Detection
- Configure Data Collectors
- Configure Call Graphs
- Configure JMX Metrics from MBeans and [Import or Export JMX Metric Configurations](#) - pertains to the **Configure JMX** permission
- Configure Memory Monitoring (Java)
- Configure End User Experience
- Configure Code Metric Information Points
- Configure Policies
- Configure Baselines
- Notification Actions
- SQL Capture Settings - pertains to the **Configure SQL Bind Variables** permission
- App Agent Node Properties

## Custom Dashboard Permissions

Permissions that can be granted at the custom dashboard level include:

- View
- Edit
- Delete

Custom dashboards are a good way to present a selected set of metrics for a "guest" user, such as an executive, who does not normally use AppDynamics. You can grant such users' permission to view custom dashboards created especially for them.

## Learn More

- [Configure Custom Roles](#)

## Access Role Configuration

- [To Access the Roles Configuration Screen](#)

### To Access the Roles Configuration Screen

1. Access the security settings configuration screen. See [Configure Authentication, User Permissions and Integrations](#).

2. Click the **Roles** tab.

The Roles configuration screen opens.

## View Predefined Roles

- [To View a Predefined Role](#)
- [Learn More](#)

You can view the predefined roles and assign them to users and groups if you have the required permissions.

If the checkbox for a permission in a role configuration is checked, the permission is granted to users with that role. If the checkbox is clear, the permission is denied.

See [Configure Roles](#) for general information about roles.

### To View a Predefined Role

1. Access the Role Configuration screen. See [Access Role Configuration](#).
  2. Click the **Roles** tab.
  3. In the left panel select the role that you want to view.
  4. In the right panel, click the **Account Level Permissions** tab to see the role's permission at the account level.
  5. Click the **Custom Dashboards** tab to view the role's permissions for viewing, editing, and deleting custom dashboards.,
  6. Click the **Application Permissions** tab to see the role's permissions at application and tier levels. You need to expand the applications to see the tier-level permissions.
    - a. Select the application or tier for which you want to view the configuration. You need to expand the applications tabs to view the tiers.
    - b. Click the Edit icon.
- The Edit Permissions window opens showing the edit permissions for the selected predefined role for the selected application or tier. Although you cannot edit these permissions, you can view which permissions have been granted and then determine whether the default role meets your needs for the selected application or tier or whether you need to configure a custom role. See [Configure Custom Roles](#).
- c. Click **Close** to close the Edit Permissions window.

## Learn More

- [Configure Roles](#)
- [Access Role Configuration](#)

## Configure Custom Roles

- Configuring Custom Roles
  - [To Create a Custom Role](#)
- Configuring Application Level Permissions
  - [To Configure the Default Application Permissions](#)

- To Configure Application Permissions
- To Configure Tier-Level Permissions
- Configuring Custom Dashboard Permissions
  - To Configure the Default Custom Dashboard Permissions
  - To Configure Permissions for Individual Dashboards
- Configuring Account Level Permissions
  - To Configure Account Level Permissions
- Modifying a Custom Role
  - To Modify a Custom Role
- Learn More

Create custom roles if you want to grant permissions to users and groups using roles that are configured differently from the predefined roles. See [View Predefined Roles](#) to see exactly which permissions are granted and denied by the predefined roles.

You can configure custom roles very finely to grant users certain permissions in a single application or even a single tier or set of tiers or custom dashboard.

See [Configure Roles](#) for information about the types of permissions that roles grant.

## Configuring Custom Roles

After you have created a custom role, select it and configure the three categories of permissions by clicking the tabs:

- Application Level Permissions
- Custom Dashboard Permissions
- Account Level Permissions

### To Create a Custom Role

1. Open the role configuration screen. See [Access Role Configuration](#).
2. Either
  - a. In the left panel, click the Add icon to create a new role.
  - b. In the right panel, enter the name of the role and an optional description.
  - c. Click **Save**.

or

  - a. Select an existing role in the left panel.
  - b. Click the Duplicate icon.
  - c. In the Duplicate Role window, overwrite the default name with your name for the new role.
  - d. Click **Duplicate**.
  - e. In the left panel, select the newly created role.
  - f. In the right panel, edit the name and description for the new role.
  - g. Click **Save**.

## Configuring Application Level Permissions

In this tab you can configure:

- the role's default application-level permissions
- the role's custom permissions for specific applications and tiers

Custom roles can be very fine-tuned at the application level and tier levels. For example, you could create an AcmeManager role with certain delete and edit permissions that apply only to the Acme bookstore application and another AcmeUser role with only view permissions or possibly also with a more limited set of edit permissions for the Acme bookstore application. You can also create roles at the tier level; for example, an InventoryManager role or an InventoryUser role with certain permissions only the Inventory tier and not for any of the other tiers in the application.

Every role has a set of default application-level permissions that are inherited by all new applications in the account. For a custom role you can reconfigure these default permissions. You can also override the inherited permissions for custom roles by reconfiguring application-level and tier-level permissions.

### To Configure the Default Application Permissions

1. With the custom role that you are configuring selected in the left panel, in the right panel click the **Application Permissions** tab.
2. To grant the role permission to create new applications, check the **Can Create Applications** check box. Otherwise leave it clear.
3. In the Default row do the following:
  - To permit users with this role to view applications check the **View** check box. To deny them view permission, clear the **View** check box.
  - To permit users with this role to delete the application check the **Delete** check box. To deny them delete permission, clear the **Delete** check box.
  - To permit users to edit AppDynamics configurations:
    - a. Click the **Edit** icon.
    - b. In the Edit Permissions window check the check boxes for the configurations that the role can perform and clear them for the configurations that it cannot perform.For information about the permissions that can be granted at the application level and tier levels, see [Application Level Permissions](#)
4. Click **OK** in the Edit Permissions window.
5. Click **Save** at the top of the pane to save the default configuration.

### To Configure Application Permissions

You can configure a custom role to have different permissions in different applications.

1. With the custom role that you are configuring selected in the left panel, in the right panel click the **Application Permissions** tab.
2. In the row for the application for which you want to configure the role's permissions, do one of the following:
  - If you want the role to inherit the default application-level permissions, select **Inherit from Default** from the dropdown menu if it is not already selected.  
or
    - If you do not want the role to inherit the default application-level permissions, select **Customized** from the dropdown menu and then follow steps 3 and 4 as described above in [To Configure the Default Permissions](#) for the application's row instead of for the Default row.
3. Repeat the previous step for every application that you want to configure.
4. Click **Save** at the top of the pane to save the configuration.

### To Configure Tier-Level Permissions

By default, a role's permissions in a tier are inherited from the role's permissions in the tier's containing application. You can override this behavior to configure a custom role to have specific permissions in different tiers.

1. In the **Application Permissions** tab, expand the application in which you want to configure the role's tier-level permissions.
2. In the row for the tier for which you want to configure the role's permissions, do one of the following:
  - If you want the role to inherit the application permissions for the tier, select **Inherit from Application** from the dropdown menu if it is not already selected.  
or
    - If you do not want the role to inherit the application permissions for the tier, select **Customized** from the dropdown menu and then follow steps 3 and 4 as described above in [To Configure the Default Permissions](#) for tier's row instead of for the Default row.
3. Repeat the previous step for every tier that you want to configure.

4. Click **Save** at the top of the pane to save the configuration.

## Configuring Custom Dashboard Permissions

In this tab you can configure:

- the role's default custom dashboard permissions
- the role's permissions for specific custom dashboards

Every dashboard inherits the default custom dashboard permissions unless you override them by configuring separate permissions for individual dashboards. For example, you could have a custom dashboard called SalesDashboard and a custom role SalesRole, and another custom dashboard called FinanceDashboard and a custom role FinanceRole. The SalesRole could be configured to have permissions in the SalesDashboard but no permissions in the FinanceDashboard and vice-versa.

### To Configure the Default Custom Dashboard Permissions

1. With the custom role that you are configuring selected in the left panel, in the right panel click the **Custom Dashboard Permissions** tab.

2. To grant the role permission to create new custom dashboards, check the **Can Create Custom Dashboards** check box. Otherwise leave it clear.

3. In the default row do the following:

- To permit users with this role to view custom dashboards, check the **View** check box. To deny them view permission, clear the **View** check box.
- To permit users with this role to edit custom dashboards, check the **Edit** check box. To deny them edit permission, clear the **Edit** check box.
- To permit users with this role to delete custom dashboards, check the **Delete** check box. To deny them delete permission, clear the **Delete** check box.

4. Click **Save** at the top of the pane to save the default configuration.

### To Configure Permissions for Individual Dashboards

You can configure a custom role to have different permissions in different custom dashboards.

1. With the custom role that you are configuring selected in the left panel, in the right panel click the **Custom Dashboard Permissions** tab.

2. In the row for the custom dashboard for which you want to configure the role's permissions:

- To permit users with this role to view the custom dashboard, check the **View** check box. To deny them view permission, clear the **View** check box.
- To permit users with this role to edit the custom dashboard, check the **Edit** check box. To deny them edit permission, clear the **Edit** check box.
- To permit users with this role to delete the custom dashboard, check the **Delete** check box. To deny them delete permission, clear the **Delete** check box.

3. Repeat the previous step for every custom dashboard that you want to configure.

4. Click **Save** at the top of the pane to save the configuration.

## Configuring Account Level Permissions

### To Configure Account Level Permissions

1. With the role selected in the left panel, in the right panel click the **Account Level Permissions** tab.

2. Check the check boxes for the tasks that can be performed by the selected custom role. see [Account Level Permissions](#) for descriptions of these permissions.

3. Clear the check boxes for the tasks that cannot be performed by this role if they are not already clear.

4. Click **Save**.

## Modifying a Custom Role

Modifying a custom role is similar to creating a new role.

### To Modify a Custom Role

1. Select the role to modify in the left panel of the role configuration screen.  
The role configuration is visible in the right panel.
2. Edit the role as you would for a creating a new custom role, as described in the preceding sections.

## Learn More

- [Configure Roles](#)

## Assign Roles to Users and Groups

- [To Assign a Role from the Role Configuration Screen](#)
- [Learn More](#)

This topic describes how to assign roles from the role configuration screen when the users and groups already exist.

You can also assign existing roles when you create users and groups from the user and group configuration screens. See [To Assign A Role to a User](#) and [To Assign A Role to a Group](#).

See [Configure Roles](#) for general information about configuring roles.

### To Assign a Role from the Role Configuration Screen

1. Access the Role Configuration screen. See [Access Role Configuration](#).
2. In the left Role Name panel select the Role that you want to assign.
3. In the right panel, click the **Users and Groups with this Role** tab.
4. To assign a role to a user or withhold a role from a user:
  - a. Locate the user or users to whom you want to assign the selected role from the Users list. You can enter a string in the filter field to locate a specific user.
  - b. Check the Member checkboxes of the users to whom you want to assign the role. A single user can be assigned multiple roles. You can click **Select All** to check all the users.
  - b. Clear the Member check boxes of the users from whom you want to withhold the role. You can click **Unselect All** to clear all the users.
5. To assign a role to a group or withhold a role from a group:
  - a. Locate the group or groups in the Groups list. You can enter a string in the filter field to locate a specific group.
  - b. Check the Member check boxes of the groups to which you want to assign the role. A single group can be assigned multiple roles. You can click **Select All** to check all the groups.
  - b. Clear the Member check boxes of the groups from whom you want to withhold the role. You can click **Unselect All** to clear all the groups.

## Learn More

- [Access Role Configuration](#)
- [Configure Groups](#)
- [Configure Users](#)

## Configure Integrations

- To configure integrations

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

## To configure integrations

1. Click the Setup menu in the upper right section of the screen.
2. From the drop-down menu, click **Administration**.
3. Click the **Integration** tab.
4. 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.
5. Check the Enabled check box to enable the integration. Clear this check box to disable the integration.
6. Provide the product-specific information in the text fields.
7. Click **Save**.

# Export and Import Business Application Configurations

- Copying Business Application Configurations
  - To export an application configuration
  - To import configuration information
- Learn More

This topic describes the how to use the import and export functionality to copy configurations from one AppDynamics business application to a new business application.

## Copying Business Application Configurations

You can export a business application configuration and import it as a new business application.

The two applications can be on the same or different Controllers. If on different Controllers they must be using the same major version of AppDynamics.

The configuration information is in an XML file that includes:

- Snapshot collection settings
- Call graph settings
- SLA configurations
- Error configuration
- Stall configuration and Business Transaction thresholds
- HTTP and SQL Data gatherer settings
- Tier information
- Custom entry point configuration for Business Transactions
- Memory configurations
- Metric baselines

The configuration information does not include data for:

- Events
- Policies
- Alerts
- Metrics

### To export an application configuration

1. In the upper right menu bar, click **Applications -> Export Application**.
2. In the Application Export window, select the application to export from the drop down list.
3. Click **Export**.  
AppDynamics creates an XML file containing application configurations.

**Tip:** If you receive 401 error while exporting the configuration, use "@customer1" after the username for the export application option to work. Customer1 is the default account name.

## To import configuration information

 **Note:** Application import will not work over HTTPS if you use a self-signed SSL certificate. Use a customer certificate. See Controller SSL and Certificates.

1. In the upper right menu bar, click **Applications -> Import Application**.
3. Select the XML file that was exported.
4. Enter a name for the application.
5. Click **Import**.

AppDynamics creates a new application with the copied configuration.

## Learn More

- Import or Export JMX Metric Configurations
- Hierarchical Configuration Model
- Upgrade the App Agent for Java
- Upgrade the App Agent for .NET

# Configure File Descriptor Limits on Linux

- Limiting the Number of File Descriptors
  - To check the limits for number of file descriptors
  - To configure limits on a per-user basis in the limits.conf file
  - To configure limits for number of file descriptors in the profile
- Learn More

This topic explains the importance of checking file descriptor limits and provides the steps to modify the limit for Controller installations on Linux.

## Limiting the Number of File Descriptors

If you do not set the limits for number of file descriptors, AppDynamics may produce warnings such as:

- Warning in database log: "Could not increase number of max\_open\_files to more than xxxx".
- Warning in server log: "Cannot allocate more connections".

These warnings indicate that the Controller database is attempting to open more file descriptors than is permitted by the operating system. This issue can affect product integrity in larger Controller installations.

AppDynamics recommends that the maximum number of file descriptors be set to **65535** for installations in Linux environments.

## To check the limits for number of file descriptors

As the root user, execute following command:

```
ulimit -S -n
```

If this value is less than 65535, change it to 65535 either in the profile or limits.conf file.

On Fedora as well as Ubuntu (and probably other modern Linux distributions) using /etc/security/limits.conf is the preferred method. You should put the ulimit command in /etc/profile only if the /etc/security/limits.conf does not exist.

## To configure limits on a per-user basis in the limits.conf file

1. Open the limits.conf file for edit:

```
/etc/security/limits.conf
```

2. Add the following lines:

```
appdynamics    hard    nofile 65535
appdynamics    soft    nofile 65535
```

Where "appdynamics" is the username of the Linux user who runs the Controller.

3. Enable these limits by editing:

```
/etc/pam.d/common-session
```

4. Add the line:

```
session required pam_limits.so
```

5. When you log in again as the "appdynamics" user, the limits should apply.

## To configure limits for number of file descriptors in the profile

1. As the root user, add the following entry to the /etc/profile:

```
ulimit -n 65535
```

2. Reboot the host server and check the limit again.

## Learn More

- Install the Controller on Linux
- Controller System Requirements

## Configure Swappiness on Linux

- The Swappiness Parameter in Linux
  - To configure swappiness
  - Learn More

## The Swappiness Parameter in Linux

The swappiness parameter controls how often the Linux kernel moves processes out of physical memory and onto the swap disk. Because disks are much slower than RAM, this can lead to slower response times for system and applications if processes are too aggressively moved out of memory.

The default value is usually "60". For better AppDynamics performance, the recommended value is "0" (zero).

### To configure swappiness

1. Check the current value for swappiness.

```
/sbin/sysctl -a | grep swappiness
```

2. Configure the swappiness parameter.

```
echo 0 > /proc/sys/vm/swappiness
```

Edit the /etc/sysctl.conf file and add following line:

```
vm.swappiness = 0
```

### Learn More

- [Install the Controller on Linux](#)

## Configure the Machine Agent to Automatically Start on Linux

- To create a start script for the Machine Agent
- [Learn More](#)
- To create a start script for the Machine Agent
- [Learn More](#)

You can configure the Machine Agent to run automatically when the machine starts.

### To create a start script for the Machine Agent

1. Create a script file as shown below:

```

#!/bin/sh
# -----
# Typical AppDynamics Machine Agent Startup
# -----

#CHANGE ME: Set to the agent's install directory
JAVA=java

#CHANGE ME: Set to the agent's install directory
AGENT_HOME=/opt/appdynamics/machineagent
AGENT="$AGENT_HOME/machineagent.jar"

# Agent Options
# Uncomment and make available as needed
# -Dappdynamics.agent.applicationName : application that the agent participates in
# -Dappdynamics.agent.logging.dir : directory to put logs (agent "user" must have write
permissions
# -Dmetric.http.listener=true | false : open a kill port
# -Dmetric.http.listener.port : the port to send kill messages

AGENT_OPTIONS=
#AGENT_OPTIONS="$AGENT_OPTIONS -Dappdynamics.agent.applicationName=<application-name>"
#AGENT_OPTIONS="$AGENT_OPTIONS -Dappdynamics.agent.logging.dir="
#AGENT_OPTIONS="$AGENT_OPTIONS -Dmetric.http.listener=true | false"
#AGENT_OPTIONS="$AGENT_OPTIONS -Dmetric.http.listener.port=<port>"

#Consolidation Step - allows you to easily add or remove the agent
nohup $JAVA $AGENT_OPTIONS -jar $AGENT &

```

2. Add the script file to the rc utility

For Red Hat and most Linux Operating Systems	For Ubuntu and Debian Operating Systems
<p>Add the script file created in Step-1 in /etc/rc.local  <b>For example:</b>  echo '/path/to/startup/script.sh start' &gt;&gt; /etc/rc.local</p>	<p>Add the script file created in Step-1 as shown below:  <b>For example:</b>  update-rc.d -f script.sh start 99 2 3 4 5.  where</p> <ul style="list-style-type: none"> <li>• start is the argument given to the script (start, stop).</li> <li>• 99 is the start order of the script (1 = first one, 99= last one)</li> <li>• 2 3 4 5 are the runlevels to start</li> </ul> <p> <b>IMPORTANT:</b> Do not forget to add dot(.) at the end.</p>

## Learn More

- [Install the Machine Agent](#)
- [Machine Agent Configuration Properties](#)

# 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 GPMERVE, 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&id=8D0460
% users                =
http://192.86.32.72:8803/gpm/perform.xml?resource=%22,SVSCPLEX,SYSPLEX%22&id=8D0D50
% using for i/o by MVS image    =
http://192.86.32.72:8803/gpm/perform.xml?resource=%22,SVSCPLEX,SYSPLEX%22&id=8D1DA0
% CSA utilization by MVS image =
http://192.86.32.72:8803/gpm/perform.xml?resource=%22,SVSCPLEX,SYSPLEX%22&id=8D2410
% users by MVS image        =
http://192.86.32.72:8803/gpm/perform.xml?resource=%22,SVSCPLEX,SYSPLEX%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

# Machine Agent Configuration Properties

- Where to Configure Machine Agent Properties
- Example Machine Agent controller-info.xml File
- Example Startup Configuration Using System Properties
- Machine Agent Properties
  - Agent-Controller Communication Properties
    - Controller Host Property
    - Controller Port Property
  - Machine Agent Identification Properties
    - Application Name Property
    - Tier Name Property
    - Node Name Property
  - Multi-Tenant Mode Properties
    - Account Name Property
    - Account Access Key Property
  - Proxy Properties for the Controller
    - Proxy Host Property
    - Proxy Port Property
  - Other Properties
    - Controller SSL Enabled Property
    - Enable Orchestration Property
    - Force Agent Registration Property
    - Unique Host ID Property
- Learn More

## Where to Configure Machine Agent Properties

You can configure Machine Agent properties:

- in the controller-info.xml file located in the <Machine\_Agent\_Installation\_Directory>/conf directory
- in the system properties (-D options) in the JVM start-up script

The system properties override the settings in the controller-info.xml file.

## Example Machine Agent controller-info.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<controller-info>

<controller-host>192.10.10.10</controller-host>

<controller-port>8090</controller-port>

<controller-ssl-enabled>false</controller-ssl-enabled>

<enable-orchestration>false</enable-orchestration>

<account-name></account-name>
<account-access-key></account-access-key>

<application-name></application-name>
<tier-name></tier-name>
<node-name></node-name>

<force-agent-registration>false</force-agent-registration>

</controller-info>
```

## Example Startup Configuration Using System Properties

A bash example:

```
-Dappdynamics.controller.hostName=192.168.1.20 -Dappdynamics.controller.port=8090
-Dappdynamics.agent.applicationName=ACMEOnline -Dappdynamics.agent.tierName=Inventory
-Dappdynamics.agent.nodeName=inventory1 org.tomcat.TomcatServer
```

## Machine Agent Properties

This section describes the Machine Agent configuration properties, including their controller-info-xml elements and their system property options.

## Agent-Controller Communication Properties

### Controller Host Property

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

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

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

**Type:** String

**Default:** None

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

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

## Controller Port Property

**Description:** This is the HTTP(S) port of the AppDynamics Controller. This is the same port that you use to access the AppDynamics browser-based user interface. If the Controller SSL Enabled property is set to true, specify the HTTPS port of the Controller; otherwise specify the HTTP port. See [Controller SSL Enabled Property](#).

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

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

**Type:** Positive Integer

**Default:** The default values are 8090 for HTTP and 8181 for HTTPS.

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

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

## Machine Agent Identification Properties

If the machine agent is installed on a machine that does not have an App Server agent, configure the application name, tier name and the node name. Otherwise these configurations are not required for the machine agent.

### Application Name Property

**Description:** This is the name of the logical business application that this JVM node belongs to. Note that this is not the deployment name(ear/war/jar) on the application server.

If a business application of the configured name does not exist, it is created automatically.

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

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

**Type:** String

**Defaults:** None

**Required:** If a registered app server agent is already installed on the same host as this machine agent, this configuration is not required.

### Tier Name Property

**Description:** This is the name of the logical tier that this JVM node belongs to. Note that this is not the deployment name (ear/war/jar) on the application server.

If a tier of the configured name does not exist, it is created automatically.

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

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

**Type:** String

**Defaults:** None

**Required:** If a registered app server agent is already installed on the same host as this machine agent, this configuration is not required.

### Node Name Property

**Description:** This is the name of the JVM node.

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

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

**Type:** String

**Defaults:** None

**Required:** If a registered app server agent is already installed on the same host as this machine agent, this configuration is not required.

## Multi-Tenant Mode Properties

If the AppDynamics Controller is running in multi-tenant mode or if you are using the AppDynamics SaaS Controller, specify the account name and account access key for this agent to authenticate with the Controller.

If the Controller is running in single-tenant mode (the default) there is no need to configure these values.

### Account Name Property

**Description:** This is the account name used to authenticate with the Controller.

If you are using the AppDynamics SaaS Controller, the Account Name is provided in the Welcome email sent by AppDynamics.

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

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

**Type:** String

**Default:** None

**Required:** Yes for AppDynamics SaaS Controller and other multi-tenant users; no for single-tenant users.

### Account Access Key Property

**Description:** This is the account access key used to authenticate with the Controller.

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

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

**Type:** String

**Default:** None

**Required:** Yes for AppDynamics SaaS Controller and other multi-tenant users; no for single-tenant users.

## Proxy Properties for the Controller

These properties route data to the Controller through a proxy.

### Proxy Host Property

**Description:** This is the proxy host name or IP address.

**Element in controller-info.xml:** Not applicable

**System Property:** -Dappdynamics.http.proxyHost

**Type:** String

**Default:** None

**Required:** No

## Proxy Port Property

**Description:** This is the proxy HTTP(S) port.

**Element in controller-info.xml:** Not applicable

**System Property:** -Dappdynamics.http.proxyPort

**Type:** Positive Integer

**Default:** None

**Required:** No

## Other Properties

### Controller SSL Enabled Property

**Description:** This property specifies whether the agent should use SSL (HTTPS) to connect to the Controller. If SSL Enabled is true, set the Controller Port property to the HTTPS port of the Controller. See [Controller Port Property](#).

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

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

**Type:** Boolean

**Default:** False

**Required:** No

### Enable Orchestration Property

**Description:** When set to true, this property enables machine agent workflow task execution.

It also enables auto-detection of the controller host and port when the app server is a compute cloud instance created by an AppDynamics orchestration workflow. In a cloud compute environment, auto-detection is necessary for the Create Machine tasks in the workflow to run correctly.

See [Controller Host Property](#) and [Controller Port Property](#).

The machine agent polls for task executions only when orchestration is enabled.

If the host machine on which this agent resides is not created through AppDynamics workflow orchestration, this property should be set to false.

**Element in controller-info.xml:** <enable-orchestration>

**System Property:** Not applicable

**Type:** Boolean

**Default:** False

**Required:** No

### Force Agent Registration Property

**Description:** Set to true only under the following conditions:

- The Agent has been moved to a new application and/or tier from the UI and
- You want to override that move by specifying a new application name and/or tier name in the agent configuration.

If there is already a registered app server agent installed on the same host as this machine agent, this override does not work. If you want to override the UI in this case, you must force the agent registration change from the app server agent configuration.

**Element in controller-info.xml:** <force-agent-registration>

**System Property:** Not applicable

**Type:** Boolean

**Default:** False

**Required:** No

## Unique Host ID Property

**Description:** This property logically partitions a single physical host or virtual machine.

You can use the unique host id when you want to use the same node name for multiple nodes on the same physical machine.

Set the value to a string that is unique across the entire managed infrastructure. The string may not contain any spaces.

If this property is set on the machine agent, it must be set on the app agent as well.

Note that if more than one app agent is running on the host, to see machine agent metrics it is necessary to run a new machine agent instance every time you specify a different unique host id on that host.

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

**Type:** String

**Default:** None

**Required:** No

## Learn More

- [Install the Machine Agent](#)
- [Machine Agent Install and Admin FAQ](#)

# Configure Multiple Machine Agents for One Machine

- [Configuring Multiple Machine Agents on Single Machine \(Java Only\)](#)
  - [To configure two machine agents for two applications running on the same machine](#)
- [Sample Configuration](#)
- [Learn More](#)

## Configuring Multiple Machine Agents on Single Machine (Java Only)

If you have different applications running on the same machine, to get hardware metrics for each application, run separate machine agents on the same machine.

To do this, create multiple copies of the machine agent. Then configure each machine agent and each app agent pair to use the same application name and Unique Host ID. The Unique Host ID makes it appear to the Controller that the application is running on different machines. See [Application Name Property](#) and [Unique Host ID Property](#).

The following instructions assume two applications and two machine agents on a single machine, but they can be interpolated to cover more than two.

## To configure two machine agents for two applications running on the same machine

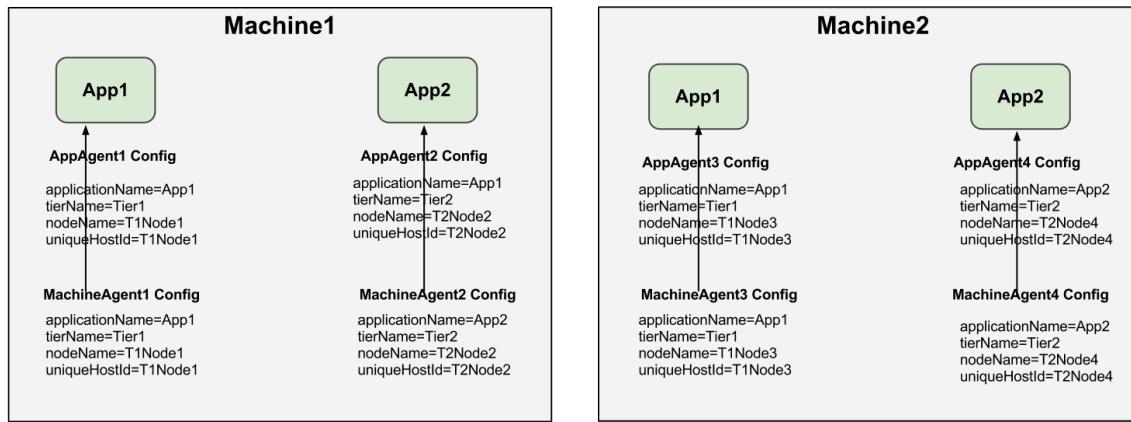
1. Download two copies of the machine agent, one for each application.
2. Assign the machine agents different names, for example: "MachineAgent1" and "MachineAgent2". If there are custom scripts running on the machine agents, they must not use the same resources.
3. Configure the first application/machine agent pair for the first application:
  - Delete the app agent node from the Controller UI.
  - Configure the applicationName and UniqueHostID properties for the app agent.
  - Configure the applicationName and UniqueHostID properties for the machine agent using the same application name and unique host id values that you used for the app agent configuration.
4. Configure the second application/machine agent pair for the second application:
  - Delete the app agent node from the Controller UI.
  - Configure the applicationName and UniqueHostID properties for the app agent. These values must be different from the values used for the first application.
  - Configure the applicationName and UniqueHostID properties for the machine agent using the same application name and unique host id values that you used above for the app agent configuration.
5. Restart all the JVMs.

## Sample Configuration

The following sample configures two physical machines.

Machine1 runs App1, which is instrumented with one app agent (AppAgent1) and one machine agent (MachineAgent1). Machine1 also runs App2, which is instrumented with one app agent (AppAgent2) and one machine agent (MachineAgent2).

Machine2 runs App1, which is instrumented with one app agent (AppAgent3) and one machine agent (MachineAgent3). Machine2 also runs App2 is instrumented with one app agent (AppAgent4) and one machine agent (MachineAgent4).



In the Controller, separate metrics are reported for:

Machine1\_App1  
Machine1\_App2  
Machine2\_App1  
Machine2\_App2

## Learn More

- App Agent for Java Configuration Properties
- Machine Agent Configuration Properties

# Configure an On-Premise Controller

This section describes how to configure the Controller for best results.

## Controller Licenses

- The License File for On-Premise Installations
  - To add the Controller license file on Windows
  - To add the Controller license file on Linux

This topic discusses how to manage Controller licenses.

### The License File for On-Premise Installations

Download the license file, license.lic, from the Welcome email that you receive when you sign up for the trial or when you purchased AppDynamics.

You do not need to deploy the license file for SaaS installations.

#### To add the Controller license file on Windows

1. Download the license file.
2. Copy and paste this license.lic file to the <Controller\_Installation\_Directory>.

#### To add the Controller license file on Linux

1. Download the license file.
2. Use the cp command to copy the downloaded license file to the Controller installation directory.

```
>> cp license.lic <Controller_Installation_Directory>
```

3. Wait at least 30 seconds for the Controller to pick up the new license file.

## Controller Tenant Mode

ays

- Tenant Mode and the Controller Accounts with Access to Business Applications
  - Switching Between Single- and Multi-Tenant Mode
    - To switch between single- and multi-tenant mode
  - Accounts for Multi-Tenant Mode
    - To create accounts in multi-tenant mode
    - Agent-Controller Communication Settings
      - To update agents with the new Controller Account information
  - Learn More

### Tenant Mode and the Controller Accounts with Access to Business Applications

You can configure the AppDynamics Controller in either single-tenant (single account) or multi-tenant (multiple account) modes. By

default, the Controller runs in single-tenant mode.

The differences are:

#### In single-tenant mode

- There is only one account (tenant) in the Controller system.
- All users are part of this single built-in account. Similarly, all Applications are part of that single built-in account.
- All the users will have access to all monitored Applications in this mode.
- AppDynamics recommends single-tenant mode for most installations.

#### In multi-tenant mode

- You can create multiple accounts (tenants).
- Each account will have its own set of users and Applications. The users in an account will only have access to Applications in that account.
- Multi-tenant mode allows secure partitioning of users and data among different parts of your organization.
- If you install the Controller in multi-tenant mode, the agents must specify additional information about the account to which they connect. For details see [App Agent for Java Configuration Properties](#), [App Agent for .NET Configuration Properties](#), and [Machine Agent Configuration Properties](#).

## Switching Between Single- and Multi-Tenant Mode

You can choose the tenancy mode when you run the Controller installer. You can switch between these two modes after you have installed the Controller.

 Note: these instructions are applicable only if you have installed the Controller on-premise.

### To switch between single- and multi-tenant mode

1. Use the following URL for accessing the account management information.

```
http://<controller-installer-host>:<port>/controller/admin.html
```

 Note: You have to be logged in as the root user. The default password for the root user is "changeme", for both single- and multi-tenant mode.

2. Click **Controller Settings**.

3. If you want to switch from single to multi-tenant mode, set the value for the multi-tenant.controller property to "true".

4. Save the settings.

5. Log out and refresh your browser.

## Accounts for Multi-Tenant Mode

In multi-tenant mode you create accounts using the "Accounts" setting.

### To create accounts in multi-tenant mode

1. Use the following URL to access the account management information:

```
http://<controller-installer-host>:<port>/controller/admin.html
```

 Note: You have to be logged in as the root user. The default password for the root user is "changeme", for both single- and multi-tenant mode.

2. Click **Accounts**.

If you are switching from single to multi-tenant mode, you will see the default accounts used for single-tenant mode.

This window also provides information on the number of licenses provisioned for the agents in your environment.

3. Click **Add** (the + icon).

4. Provide the details about the Account.

Users will log into the AppDynamics UI using this Account Name.

5. Create credentials for the Controller administrator.

The user with administrator privileges can then create users for that Account. For details see [Controller Users](#).

6. Click **Create account**.

## Agent-Controller Communication Settings

When you change tenant modes, you must also update the app server and machine agents with the new Account information.

### To update agents with the new Controller Account information

For details see [App Agent for Java Configuration Properties](#), [App Agent for .NET Configuration Properties](#), and [Machine Agent Configuration Properties](#).

## Learn More

- [Controller Users](#)
- [App Agent for Java Configuration Properties](#)
- [App Agent for .NET Configuration Properties](#)
- [Machine Agent Configuration Properties](#)
- [Install the App Agent for Java](#)
- [Install the App Agent for .NET](#)
- [Install the Machine Agent](#)

## Controller Port Settings

- [Controller Ports](#)
- [Changing the Controller Port for On-Premise Installations](#)
  - [Reinstalling the Controller with New Port Settings](#)
    - To reinstall the Controller with new port settings
  - [Editing Controller Port Configurations](#)
    - To edit configuration files and change the Controller port settings

## Controller Ports

The Controller requires the following port connection settings in the configuration files:

Port Name	Default
Application server primary port	Port 8090 is used for HTTP and port 8181 is used for HTTPS.
Database server port	3388
Application server admin port	4848
Application server JMS port	7676
Application server IIOP port	3700

## Changing the Controller Port for On-Premise Installations

If you have installed the Controller on-premise, you can change the port settings using either of the following procedures:

- Reinstalling the Controller
- or
- Editing Controller Port Configurations

## Reinstalling the Controller with New Port Settings

Use the reinstall procedure if you want to modify connection settings without editing configuration files.

### To reinstall the Controller with new port settings

1. Shut down the Controller. For instructions see [Start or Stop the Controller](#).
2. Create a complete backup of the Controller installation directory and all of its sub-directories.
3. Re-install the Controller in the same installation directory as your original Controller and choose the new port settings while running the installer. For instructions see [Install the Controller on Linux](#) or [Install the Controller on Windows](#).

 This step will be treated as an upgrade by the installer for the Controller.

## Editing Controller Port Configurations

Use this procedure to manually edit the port connection setting without reinstalling the Controller.

### To edit configuration files and change the Controller port settings

1. Shut down the Controller. For instructions see: [Start or Stop the Controller](#).

 **Important:** Shut down the Controller before making edits to configuration files. AppDynamics stops collecting data when you shut down the Controller and resumes data collection after the Controller is restarted.

2. Modify the domain.xml file located at <Controller-Installation-Directory>/appserver/domains/domain1/config. Set each of the Controller port settings:

- Change the application server primary port.
  - Find the <http-listener> element that has the attribute id="http-listener-1" and set the value of this port to the new port setting.
  - Save your changes.
- Change the database server port.
  - Find the <jdbc-connection-pool> element that has the attribute: name="controller\_mysql\_pool".
  - Find the <property> element within <jdbc-connection-pool> that has name="portNumber".
  - Set the "value" attribute within the <property> element to the new database server port setting. Save your changes.

3. Open the config.properties file located at <Controller-Installation-Directory>/appserver/domains/domain1/imq/instances/imqbroker/props.

- Set the "imp.persist.jdbc.mysql.property.url" variable (the JDBC connection string) to your new port setting.
- Save your changes.

4. Open the db.cnf file located at <Controller-Installation-Directory>/db.

- Set the "port=" variable to your new port setting.
- Save your changes.

5. Open the controller.bat (.sh) file located at <Controller-Installation-Directory>\bin.

- Change the "DB\_PORT" variable to your new port setting.
- Save your changes.

6. Open the domain.xml file located at <Controller-Installation-Directory>/appserver/domains/domain1/config.

- Find the <http-listener> element that has following attribute: id="admin-listener".
- Set the port (port=) value to your new port setting.
- Save your changes.

7. Open the asadminenv.conf file located at <Controller-Install-Directory>/appserver/config.
  - Change the "AS\_ADMIN\_PORT" variable to your new port setting.
  - Save your changes.
8. Open the domain.xml file located at <Controller-Installation-Directory>/appserver/domains/domain1/config.
  - Find the <jms-host> element that has the attribute name="default\_JMS\_host".
  - Set the value of "port=" variable to your new port setting.
  - Save your changes.
9. Open the domain.xml file located at <Controller-Installation-Directory>/appserver/domains/domain1/config. Find the <iiop-listener> element that has the attribute: id="orb-listener-1".
  - Set the value of the "port=" variable to your new port setting.
  - Save your changes.
10. Restart the Controller. For instructions see [Start or Stop the Controller](#).

## Controller SSL and Certificates

- [Using SSL with the Controller](#)
  - To install a trusted certificate for the Controller

### Using SSL with the Controller

By Default, the Controller ships with a self-signed certificate, which is not recommended for use in production. We recommend that you replace this certificate with a proper CA signed certificate.

#### To install a trusted certificate for the Controller

1. Generate a key pair.

```
keytool -genkey -keyalg rsa -keystore keystore.jks -alias slas
```

 The default keystore password is "changeit". The alias is "slas".

2. Change the password.

```
<Controller_Installation_Directory>/appserver/bin/asadmin
change-master-password --savemasterpassword=true
```

3. Generate a Certificate Signing Request (CSR).

```
keytool -certreq -alias slas -file controller-ca.csr -keystore keystore.jks
```

4. Make a service request to a trusted Certificate Authority (CA) to issue a valid certificate.

5. Import the root and intermediate certificates into the keystore.jks: keytool

```
keytool -import -alias slas -file <CA-signed-cert>.cert -keystore keystore.jks
```

6. Copy the keystore.jks into the config directory for the Controller located at <Controller\_Installation-Directory>/appserver/domains/domain1/config.

# Controller Logs

- Log Files Generated by the Controller
  - To change the location of the Controller log directory
- Debug Mode Logging For the Controller
  - To configure logging levels for the Controller
- Learn More

## Log Files Generated by the Controller

The Controller creates two log files in the <Controller\_Installation\_Directory>/logs directory:

- database.log
- server.log

### To change the location of the Controller log directory

1. Stop the Controller and its database. See [Start or Stop the Controller and Controller Database Scripts](#).
2. Open the <Controller\_Installation\_Directory>/db/db.cnf file.
3. Set the value of the log-error property to the new location of the database.log file.
4. Save the db.cnf file.
5. Open the <Controller\_Installation\_Directory>/appserver/domains/domain1/config/domain.xml file.
6. Locate the <log-service> element.
7. Set the value of the file attribute to the new location of the server.log file.
8. Copy (or move) the pre-existing logs directory (<Controller\_Installation\_Directory>/logs) to the new location.
- 9 Start the Controller. See [Start or Stop the Controller](#).
10. Verify that the database.log and server.log are being written to the new locations.

## Debug Mode Logging For the Controller

Debug logs provide information about possible errors in Controller operations.

### To configure logging levels for the Controller

1. Access the Controller application server.
  - Open a browser at the following URL:

```
http://<Controller-Host>:4848
```

The administrative console for the Controller Application Server opens. Port 4848 is the default port number for the Controller application server.

To log in to the Controller application server administrative console, use "admin" as the username and the password that is located in the <controller\_install\_directory>/.passwordfile file.

2. Go to "Application Server" present in the left hand side tree of the administrative console.
3. Click **Logging -> Logging Levels**.
4. Click **Additional Properties**.
5. Modify only those elements that start with "com.singularity". INFO is the default logging mode.

To enable debug mode logging for any component, change the logging level from INFO to "FINE" mode.

The screenshot shows the Sun GlassFish Enterprise Server v2.1 Administration Console. The navigation bar at the top includes 'Home', 'Version', 'User: admin | Domain: domain1 | Server: localhost'. The main title is 'Sun GlassFish™ Enterprise Server v2.1'. On the left, under 'Common Tasks', there are links for 'Registration', 'Application Server' (which is highlighted with a yellow circle labeled '1'), and 'Applications'. A large blue arrow points from the 'Application Server' link down to the 'Additional Properties' table. The 'Application Server' tab is selected, showing sub-tabs 'General', 'JVM Settings', and 'Logging' (which is highlighted with a yellow circle labeled '2'). Below these tabs is another set of sub-tabs 'General' and 'Log Levels' (with 'Log Levels' highlighted with a yellow circle labeled '3'). The main content area displays a table titled 'Additional Properties (17)' with columns 'Name' and 'Value'. The properties listed are:

	Name	Value
<input type="checkbox"/>	com.singularity.ORCHESTRATION.ENGIN	INFO
<input type="checkbox"/>	com.singularity.INCIDENTS.WRITE	INFO
<input type="checkbox"/>	com.singularity.ORCHESTRATION.AGENT	INFO
<input type="checkbox"/>	com.singularity.INCIDENTS.READ	INFO
<input type="checkbox"/>	com.singularity.EVENTS.WRITE	INFO
<input type="checkbox"/>	com.singularity.SNAPSHOTS.WRITE	INFO
<input type="checkbox"/>	com.singularity.SNAPSHOTS.READ	INFO
<input type="checkbox"/>	com.singularity.RULES.PROCESSING	INFO
<input type="checkbox"/>	com.singularity.IPS	INFO
<input type="checkbox"/>	com.singularity	INFO
<input type="checkbox"/>	com.singularity.AGENT	INFO
<input type="checkbox"/>	com.singularity.BTS	INFO
<input type="checkbox"/>	com.singularity.METRICS.READ	INFO
<input type="checkbox"/>	com.singularity.METRICS.WRITE	INFO
<input type="checkbox"/>	com.singularity.EVENTS.READ	INFO
<input type="checkbox"/>	org.hibernate.engine.StatefulPersistence	SEVERE
<input type="checkbox"/>	javax.enterprise.system.stream.out	WARNING

You can control logging information for following components:

- Orchestration
- Incidents
- Events
- Snapshots
- Rules
- AGENT
- Business Transactions (BTS)
- Metrics
- Information Points (IPS)

## Learn More

- Start or Stop the Controller
- Controller Disk Space and the Database
- Controller Database Scripts

## Controller Performance

- Monitoring Controller Performance
  - To monitor heap usage
  - To check the server.log for any errors
- Performance Issues
  - To troubleshoot Controller performance issues
  - To troubleshoot Controller issues

## Monitoring Controller Performance

To monitor Controller performance:

- Monitor heap usage
- Review the server log file

### To monitor heap usage

- **On a Windows machine:** Use the Task Manager to measure the memory usage for the Controller.
- **On a Linux machine:** Use the **top** command to get statistics for the memory data.

```
ps -elf (expect to see a "java" process and a "mysql" process)

top (expect to see java and mysql with cpu greater than 0)
```

### To check the server.log for any errors

1. Open a console and navigate to the <Controller\_Installation\_Directory>/logs/server.log file.
2. Open the log file and look for any anomalies.

## Performance Issues

If you observe degradation in Controller performance it may be due to one of the following:

- The hardware resources for the Controller might not match the correct Controller profile.
- The Controller performance profile may be incorrectly configured.

### To troubleshoot Controller performance issues

1. Confirm that the hardware matches the Controller profile you use. For details see [Controller System Requirements](#).
2. Confirm that your disk performance matches the recommended thresholds for minimum disk performance. For details see [Controller System Requirements](#).
3. Conform that the Java SDK version is exactly the same as the Java version on the Controller. To display the version of Java used by the Controller:
  - Open the command line utility.
  - Go to <Controller\_Installation\_Directory>/jre/bin
  - Run java -version.

## To troubleshoot Controller issues

If problems persist, contact Support. See AppDynamics Support for information about how contact Support and how to collect troubleshooting information.

## Configure the SMTP Server

- [To configure the SMTP server](#)
- [To troubleshoot notifications](#)
- [Learn More](#)

This topic describes how to configure the SMTP server for email and SMS notifications. In order for the email or SMS notifications to work, the SMTP server settings must be accurate.

### To configure the SMTP server

1. Click **Settings -> Email / SMS Configuration**.
2. Provide connection information about the SMTP host, port, connection information, etc.

**Email / SMS Configuration**

Specify the SMTP server that the AppDynamics Controller will use to send email/SMS alert notifications.

SMTP Host: [Input Field]

SMTP Port: [Input Field] (highlighted with blue border)

Use Secure Connection:  Never  TLS, if avail.  TLS  SSL

Use Secure Authentication:

Notification Header Text: [Input Field] [?](#)

SMS Carrier: [Input Field] [?](#)

---

**Authentication**

Authentication required:

Username: [Input Field]

Password: [Input Field]

Confirm Password: [Input Field]

3. Save the settings.

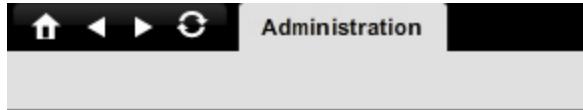
### To troubleshoot notifications

If you do not receive notifications on health rule violations, it could be because the SMTP server timeout setting is too short a period of time. To troubleshoot this problem:

1. Log in to the Admin console using the admin account:

```
http://<controller-installer-host>:<port>/controller/admin.html
```

2. Select the Controller Settings option.



## AppDynamics Administration

### Accounts

Create and Manage Accounts



3. Increase the mail.smtp.sockettimeout global configuration property. The default setting is 30 seconds.

### Learn More

- [Notification Actions](#)
- [Use a SaaS Controller#SMTP Service for SaaS](#)

## Controller High Availability

See also: [Upgrade the Controller#To upgrade Controllers configured in HA mode](#).

## Manage Controller High Availability

- [High Availability Mode](#)
- [Environment for Controller High Availability](#)
  - [TTL \(Time To Live\) of DNS A Records](#)
  - [Setting Up HA Controllers](#)
    - [Common Commands Used to Set Up Controller HA](#)
    - [To set up HA Controllers](#)
  - [Failover and Failback Procedures](#)
  - [Configuring Agents for an HA Controller](#)
- [Learn More](#)

### High Availability Mode

AppDynamics recommends that you use a [high availability cluster](#) for continued Controller operations if the server fails or becomes inoperable. You can configure the Controller in high availability (HA) mode to minimize any disruptions to Controller operations.

HA mode also helps you to avoid the complexity of performing hot backups.

## Environment for Controller High Availability

To use HA mode, you need two servers for two Controllers which are almost exactly the same. Each Controller uses its own MySQL database.

- Primary Controller, with the primary (master) MySQL database
- Secondary Controller, with the secondary (slave) MySQL database

When HA is configured, the primary (master) MySQL database replicates data to the secondary (slave) MySQL database. HA mode uses a MySQL Master-Master replication type of configuration. For more information see [Master-Master replication for MySQL](#).

Use the information at [Controller System Requirements](#) to determine the sizing and resource needs for your environment. The two servers should be functionally equivalent. If you have to use one server that is more robust than the other, use the more powerful server for the primary Controller.

### TTL (Time To Live) of DNS A Records

The Sysadmin of the DNS server should set the TTL (time to live) of the DNS A records to a short time (about 60 seconds), to more frequently refresh the IP address of the HA Controller in the agent DNS caches. This ensures that the agent gets a new name to IP translation for the HA Controller every minute or so. Otherwise the agent will continue to send messages to the failed controller until the TTL expires. If the TTL is set to the default of 84600, the agents keep using the old IP for a whole day and may not failover to the secondary controller in a timely fashion.

#### Agents and Controllers in an HA Scenario

Normally, before any failure, the App Server and Machine Agents communicate with the primary Controller. If the primary Controller becomes unavailable, the Agents start communicating with the secondary Controller.

The agents require information about both Controllers to ensure that this communication occurs.

Each Controller and MySQL database has a unique IP address, for example:

- Primary Controller has an IP address of 101.xxx.xxx.xxx
- Primary (master) MySQL Database has an IP address of 201.xxx.xxx.xxx
- Secondary Controller has an IP address of 102.xxx.xxx.xxx
- Secondary (slave) MySQL Database has an IP address of 202.xxx.xxx.xxx

When configuring the agents, AppDynamics recommends using the Controller DNS address (controller.company.com) instead of the IP address. For example, initially controller.company.com will have the IP address of 101.xxx.xxx.xxx. When the primary fails you can change the DNS to 102.xxx.xxx.xxx without changing each of the agents.

The other option is to use a load balancer that will switch Controllers with the agent knowledge. For example, the load balancer has an IP address of 103.xxx.xxx.xxx and it balances the load between 101.xxx.xxx.xxx and 102.xxx.xxx.xxx.

## Setting Up HA Controllers

The following set-up procedures cover configuring a primary and a secondary Controller and their databases.

When executing SQL statements, use the IP address for the PRIMARY-HOSTNAME and SECONDARY-HOSTNAME.

The following table lists the Linux and Windows commands used during the set-up procedures.

### Common Commands Used to Set Up Controller HA

Action	Commands for Windows	Commands for Linux
Start the Controller database	controller.bat start-db	controller.sh start-db
Log into the Controller database	controller.bat login-db	controller.sh login-db
Stop Controller database	controller.bat stop-db	controller.sh stop-db
Reset Controller database	controller.bat reset-db	controller.sh reset-db

Start Controller	startController.bat	startController.sh
Stop Controller	stopController.bat	stopController.sh

## To set up HA Controllers

1. Obtain the free license for the secondary Controller from the AppDynamics Support Team.

2. Launch the Controller installation and select HA Mode.

- On the primary Controller machine select **Primary HA Mode**.
- On the secondary Controller machine select **Secondary HA Mode**.

3. Stop both Controllers.

4. On the primary Controller machine perform the following actions:

- Start the primary Controller database and log in to the database.
- Execute the following command:

```
RESET MASTER
```

- Stop the primary Controller database.

5. Copy the <Controller\_Installation\_Directory>/db/data directory from the primary to the secondary machine.

For example, on Linux:

### Primary Controller Machine

```
tar czvf data.tgz /mnt/appdynamics2/db/data
scp data.tgz user@server2:/tmp
```

### Secondary Controller Machine

```
cd /mnt/appdynamics/db/data
rm \--fr *
tar xzvf /tmp/data.tgz
```

6. Copy the copy controller.sh or controller.bat file from the primary machine to the secondary machine.

This is to make sure that the secondary machine uses the same mysql root user password as that of primary. Otherwise you cannot log into the secondary database for setting up replication.

7. On the primary Controller machine, log in to the Controller database and execute the following command:

```
GRANT ALL ON *.* TO 'controller_repl'@'SECONDARY-HOSTNAME' IDENTIFIED BY 'controller_repl';
```

8. Perform following actions on the SECONDARY machine:

- Start the secondary Controller database.
- Reset the database.
- Log in to the database and execute the following commands:

```

RESET MASTER

GRANT ALL ON *.* TO 'controller_repl'@'PRIMARY-HOSTNAME' IDENTIFIED BY 'controller_repl';

STOP SLAVE

CHANGE MASTER TO MASTER_HOST='PRIMARY-HOSTNAME' , MASTER_USER='controller_repl' ,
MASTER_PASSWORD='controller_repl' , MASTER_PORT=YOUR-MYSQL-PORT# ;

START SLAVE

```

 DO NOT start the Controller yet.

9. On the primary Controller machine, execute following commands:

```

STOP SLAVE

CHANGE MASTER TO MASTER_HOST='SECONDARY-HOSTNAME' , MASTER_USER='controller_repl' ,
MASTER_PASSWORD='controller_repl' , MASTER_PORT=YOUR-MYSQL-PORT# ;

START SLAVE

```

10. Verify HA mode setup.

- On each machine, log into the Controller database.
- Execute the following command to display any set-up errors:

```
SHOW SLAVE STATUS\G
```

11. Start the primary Controller.

12. Set the appserver.mode property to "active" on the primary Controller.

```
http://<controller-host>:<controller-port>/controller/changeappservermode?activate=true
```

The Controller should return the message:

```
200: "App Server started in ACTIVE mode..."
```

13. Start the secondary Controller.

14. Set the appserver.mode property to "passive" on the secondary Controller.

```
http://<controller-host>:<controller-port>/controller/changeappservermode?activate=false
```

The Controller should return the message:

```
200: "App Server moved to PASSIVE mode..."
```

 Previous documentation instructed that you do not need to start the secondary Controller. This has changed and it is OK to run both Controllers.

## Failover and Fallback Procedures

If the Controller crashes on the primary machine, follow these two processes to recover:

1. Failover Process: use the appserver.mode property to switch operations to the secondary Controller machine.
2. Fallback Process: once it is recovered and stable, switch operations back to the primary Controller machine

For details see [Controller HA Failover and Failback Process](#).

## Configuring Agents for an HA Controller

When configuring the agents for a Controller in HA mode, you set the <controller-host> properties in the controller-info.xml (Linux) or Web.config (.NET) files for an App Agent and in the controller-info.xml file for a Machine Agent.

As previously discussed, use the DNS name for the <controller-host> property. For example:

```
<controller-host>controller.company.com</controller-host>
```

For more information see [Install and Upgrade AppDynamics](#).

## Learn More

- [Controller Data Backup and Restore](#)
- [App Agent for Java Configuration Properties](#)
- [App Agent for .NET Configuration Properties](#)

## Common Commands Used to Set Up Controller HA

### Common Commands Used to Set Up Controller HA

Action	Commands for Windows	Commands for Linux
Start the Controller database	controller.bat start-db	controller.sh start-db
Log into the Controller database	controller.bat login-db	controller.sh login-db
Stop Controller database	controller.bat stop-db	controller.sh stop-db
Reset Controller database	controller.bat reset-db	controller.sh reset-db
Start Controller	startController.bat	startController.sh
Stop Controller	stopController.bat	stopController.sh

## Controller HA Failover and Fallback Process

- [Switching Operations between the Primary and Secondary Controllers](#)
  - [Common Commands Used to Set Up Controller HA](#)
  - [The appserver.mode Property and changeappservermode Command](#)
- [Controller Failover](#)
  - To stop data replication on the secondary Controller
  - Increasing the Time for Storing the Replicated Data
    - To increase the data storage time to 10 days
- [Controller Fallback](#)
  - To sync up the database
  - [Learn More](#)

## Switching Operations between the Primary and Secondary Controllers

If the primary Controller experiences a failure, use the process described in this topic to switch all operations to your secondary Controller on the backup machine. When the primary Controller is restored, switch back.

## Common Commands Used to Set Up Controller HA

Action	Commands for Windows	Commands for Linux
Start the Controller database	controller.bat start-db	controller.sh start-db
Log into the Controller database	controller.bat login-db	controller.sh login-db
Stop Controller database	controller.bat stop-db	controller.sh stop-db
Reset Controller database	controller.bat reset-db	controller.sh reset-db
Start Controller	startController.bat	startController.sh
Stop Controller	stopController.bat	stopController.sh

## The appserver.mode Property and changeappservermode Command

These instructions use the appserver.mode property. You must invoke this property using the system account. For example:

```
curl --user root@system:changeme http://<host>/controller/changeappservermode?activate=true)
```

## Controller Failover

### To stop data replication on the secondary Controller

1. Confirm that the secondary Controller database is running properly on your secondary machine.

- For Linux:

```
ps -aef | grep mysql
```

- For Windows:

```
tasklist /v | find "mysql"
```

2. Log in to the secondary Controller database on the secondary Controller machine and stop replicating data from the primary Controller machine. Execute following command:

```
STOP SLAVE;
```

3. Set the appserver.mode property to "true" on the secondary Controller.

```
http://<controller-host>:<controller-port>/controller/changeappservermode?activate=true
```

The Controller should return the message:

```
200: "App Server started in ACTIVE mode..."
```

## Increasing the Time for Storing the Replicated Data

**Important:** If it will take more than two days to repair and failback the primary Controller, AppDynamics recommends that you increase the number of days for which the secondary Controller stores the replicated data.

## To increase the data storage time to 10 days

1. Login to the Controller database.
2. Execute following command:

```
SET GLOBAL expire_logs_days=10;
```

## Controller Failback

Once the primary Controller machine is back up and restored to a stable state, you switch processing back to it and restore the secondary Controller to backup or passive state.

### To sync up the database

1. On the primary Controller machine, start up only the database if it is not already started by running:

```
controller.sh start-db
```

**⚠ Important:** Do not start the application server during this phase as it will assume the appserver mode it was last in, which is likely to be "active". Running two active application servers on the same dataset can lead to corruption in the data.

2. Log in to the Controller database on the primary Controller machine and execute following command:

```
START SLAVE
```

To verify that the slave is running:

```
SHOW SLAVE STATUS \G
```

**i** Wait until the value for "Seconds\_Behind\_Master" column reaches zero. This step can be 20X faster than the time since failover event. For example, if the primary Controller machine has been down for 24 hours, it may take more than one hour for the sync process.

4. Shut down the Controller and start the Controller database on the secondary machine.

5. Log in to the Controller database for the primary Controller machine and perform following actions:

- Check that there is no database corruption on the disk or in the tables. There are third-party utilities available for this or you can use MySQL's [mysqlcheck](#).

For example, to copy the data directory on Linux:

On the Secondary Controller machine:

```
tar cvf data.bak /mnt/appdynamics2/db/data
scp user@server2:/tmp
```

On the Primary Controller machine:

```
cd /mnt/appdynamics/db/data
rm --fr *
tar xvf /tmp
```

- Replace the data directory on the primary Controller machine with the one that was copied from the secondary Controller machine.

- Start the Controller database on the primary Controller machine.
7. Start the Controller database on the secondary Controller machine.
8. Reset the Controller database on the primary Controller machine.
9. On the primary Controller machine, log into the Controller database and execute following command:

```
"RESET MASTER"
```

- If you have copied data directory from the secondary Controller to the primary Controller, also invoke following command:

```
"CHANGE MASTER"
```

10. On the secondary Controller machine, log in to the Controller database and execute following commands:

```
"CHANGE MASTER TO MASTER_HOST='PRIMARY-HOSTNAME' , MASTER_USER='controller_repl' ,
MASTER_PASSWORD='controller_repl' , MASTER_PORT=YOUR-MYSQL-PORT# ;"
"START SLAVE"
```

11. Start the Controller on the primary Controller machine.
12. Set the appserver.mode property to "passive" on the secondary Controller.

```
http://<controller-host>:<controller-port>/controller/changeappservermode?activate=false
```

The Controller should return the message:

```
200: "App Server moved to PASSIVE mode..."
```

13. Set the appserver.mode property to "active" on the primary Controller.

```
http://<controller-host>:<controller-port>/controller/changeappservermode?activate=true
```

The Controller should return the message:

```
200: "App Server started in ACTIVE mode..."
```

 **Warning:** Never set both Controllers to "active" at the same time.

## Learn More

## Automating HA Controller Failover and Failback

- Failover and Failback Scripts
  - Failover Script
    - To Failover
  - Failback Script
    - To Failback
- Moving Traffic between Controllers
  - Using Virtual IP (VIP)
  - Using a Load Balancer
- Modifying and Testing the Scripts

- To Modify and Test the Scripts
- Learn More

This topic describes how to use scripts provided by AppDynamics to perform failover and fallback operations for your AppDynamics Controller. It provides a simpler alternative to the manual failover and fallback procedures described at [Controller HA Failover and Fallback Process](#).

It assumes that you have already provisioned a primary and secondary controller in High Availability (HA) mode. For information on how to do this see [Provisioning Controllers in High Availability Mode](#). It also assumes that you also have a provisioning server, which has SSH access to both controller machines, from which to run the scripts.

Failover is the operation to perform if the primary controller experiences a failure; it switches all operations to your secondary controller on the backup machine. Fallback is the operation that switches operations from the secondary controller back to the primary controller after the primary controller has been restored.

## **Failover and Fallback Scripts**

AppDynamics provides a set of scripts that you can run to perform failover and fallback. Click [failover\\_scripts.zip](#) to download the zip file containing the scripts. Unzip all the scripts into the same directory on the provisioning server.

The scripts that you run directly are:

- failover.sh: to perform failover
- fallback.sh: to perform fallback

These two scripts invoke the other scripts to perform support tasks.

These scripts do not handle switching traffic between the primary and secondary controllers. See [Moving Load Between Controllers](#) for information on how to do this.

### **Failover Script**

This script performs the following tasks:

1. Stops traffic to the primary controller.
2. Sets the primary controller to passive mode.
3. Turn off replication.
4. Records where the primary controller last stopped replicating.
5. Stops the controller on the primary machine.
6. Makes the secondary controller the active controller.
7. Directs traffic to the secondary controller.

#### **To Failover**

- Run failover.sh from the provisioning server to failover from the primary to the secondary controller.

### **Fallback Script**

This script performs the following tasks:

1. Stops traffic to the secondary controller
2. Sets the secondary controller to passive mode.
3. Turn off the database on the primary controller and lets it replicate from the secondary controller until the primary controller has all the data collected since the failover.
4. Sets up replication again.
5. Makes the primary controller the active controller so that it can receive traffic.
6. Directs traffic to the primary controller

#### **To Fallback**

- Run fallback.sh from the provisioning server to fallback from the secondary to the primary controller after the primary controller has been restored.

## **Moving Traffic between Controllers**

To move traffic between controllers, choose from the common options listed below:

- Using Virtual IP (VIP)
- Using a Load Balancer

Based on the option you choose, make the appropriate changes to the failover and fallback scripts.

### **Using Virtual IP (VIP)**

To use the VIP approach, assign a single IP which can be grabbed and released by either machine. During failover, the machine with the primary controller releases the VIP and the machine with the secondary controller takes it by binding it to eth1.

You implement this solution using the Linux ifup and ifdown commands.

### **Using a Load Balancer**

To use the load balancer approach, point agent and UI traffic to a loadbalancer IP which then redirects the traffic to the active controller

During failover, you can manually direct the loadbalancer to switch traffic to the other controller.

## **Modifying and Testing the Scripts**

### ***To Modify and Test the Scripts***

Modify the scripts as needed and then test them.

1. Edit the export entries in setup.sh with the information for your own environment.

2. Edit the failover.sh script as needed based on your failover strategy.

At the start of the script, stop traffic to the primary controller. At the end of the script, direct traffic to the secondary controller.

3. Edit the fallback.sh script as needed based on your fallback strategy. At the start of this script, stop traffic to the secondary controller. At the end of the script, direct traffic to the primary controller.

4. Test these scripts simulating a failover and fallback to make sure everything works as expected. Then just execute failover.sh and fallback.sh when needed.

### **Learn More**

- Provisioning Controllers in High Availability Mode

## **Controller High Availability FAQ**

- Q. What do I do when the database size between the primary and the secondary Controller does not match?
- Q. Why is there an error 1045?
- Learn More

This topic lists frequently asked questions about Controller high availability configurations.

### **Q. What do I do when the database size between the primary and the secondary Controller does not match?**

To troubleshoot this problem, gather information about the system and open an [AppDynamics support ticket](#). Follow these steps:

1. For each Controller, locate the <Controller-Installation-Directory>/db/db.cnf file and make a copy of each.

Send these files to the AppDynamics Support Team.

2. Capture the secondary database status.

- Log in to the secondary Controller database.
- Execute the following command on secondary Controller database and save the results:

```
SHOW SLAVE STATUS\G
```

Send these results to the AppDynamics Support Team.

3. Capture the disk usage information on each machine.

- From a console, navigate to the <Controller-Installation-Directory>/db/data directory.
- Execute the following command and save the output:

For Linux:

```
du -h
```

For Windows:

```
du -l
```

Send these results to AppDynamics Support Team.

 You can download a Windows disk usage tool from [sysinternals site](#). You can also use tools such as [WinDirStat](#) to get the disk usage statistics.

4. Capture the file system information on each machine.

- From a console, navigate to the <Controller-Installation-Directory>/db/data/controller directory.
- Execute following command and save the output:

For Linux:

```
ls -l
```

For Windows:

```
dir
```

- Execute following command and save the output:

For Linux:

```
ls -s |sort -n
```

For Windows:

```
dir /s /os
```

Send these results to AppDynamics Support Team.

#### **Q. Why is there an error 1045?**

MySQL may return a "cannot authenticate" error code 1045 if the domain name is used for the PRIMARY-HOSTNAME and SECONDARY-HOSTNAME instead of the IP address. Use the IP address.

#### **Learn More**

# Provisioning Controllers in High Availability Mode

- Required Machines
- Required Scripts
- Required AppDynamics License and Account
  - To get license keys
  - To get a monitor account
- Set Up the Machines
  - To set up the HA machines and the provisioning server
- Learn More

This topic describes how to provision a controller in high availability (HA) mode.

For successful operations of a large scale Controller AppDynamics strongly recommends that your Controller be allowed to upload its performance data to the AppDynamics monitoring system.

## Required Machines

The Controller setup for HA requires three machines:

- Primary HA Controller
- Secondary HA Controller
- Provisioning Server

## Required Scripts

The setup uses the following scripts:

- external\_provision\_controller.sh
- controller\_call.sh
- external\_Install\_controller.sh
- keyval\_21.template

Click [provision\\_controller.zip](#) to download the zip file containing these scripts.  
Unzip them in any directory on the provisioning server.

You will edit only external\_provision\_controller.sh. This script invokes the other three scripts.

## Required AppDynamics License and Account

### To get license keys

1. Send AppDynamics Support a request for a license key for your primary and secondary controller hosts. Include with your request the MAC address of the each machine. See [AppDynamics Support](#) for information on how to contact support.

AppDynamics will generate a license key for each of your controller machines and email them to you.

2. Name the license keys in a logical way, such as "primary\_license.lic" and "secondary\_license.lic" and save them in a directory on the provisioning server. Make a note of the full path of these directories, as you will need to enter them as primary\_license and secondary\_license when you edit the external\_provision\_controller.sh script.

### To get a monitor account

1. Send AppDynamics Support a request for an account on SaaS Monitor.  
AppDynamics will send you an account name and password for the SaaS Monitor.

2. Save these values.

You will need to enter them as monitor\_acctname and monitor\_pass when you edit the external\_provision\_controller.sh script.

## Set Up the Machines

### To set up the HA machines and the provisioning server

1. Allow port 80 on both HA machines to be an open outbound port through which the Controllers will send data to the AppDynamics monitoring system.

2. Set up a third machine to be the provisioning server. The machine resources can be very small, and the machine can be virtual.

The provisioning server must have SSH access with an SSH key to both HA machines.

The provisioning server must have its SSH key set up to automatically ssh into the HA machines without prompting. See [Setting Up an SSH Key for Controller Provisioning](#) if you need detailed instructions on how to do this.

3. Confirm that all three machines (provisioning server, Primary HA Controller, and Secondary HA Controller) have the following Linux commands installed and in the path:

sed, awk, scp, ssh, ls, mkdir, unzip, java (64-bit)

4. Download the latest Machine Agent from the AppDynamics download server at <http://download.appdynamics.com/> and put it in a directory of your choosing on the provisioning server. You will need to enter the full path of the machine agent installer when you edit the external\_provision\_controller.sh script.

5. Download the latest Controller installer from the AppDynamics download server at <http://download.appdynamics.com/> and put it in a directory of your choosing on the provisioning server. You will need to enter the full path of the Controller installer when you edit the external\_provision\_controller.sh script.

6. Test the three machines to confirm that you can SSH from the provisioning server to either Controller machine without being prompted.

7. Open the external\_provision\_controller.sh script that you stored on the provisioning server and change the variables based on the machine IP numbers, machine hostnames, directories, users, etc.

8. Run the ./external\_provision\_controller.sh script to install the Controller in HA mode on the provisioning server.

9. Verify that the Machine Agent on the provisioning server will restart after any future reboot by adding the following line in /etc/rc.local:

```
nohup <path_to_64bit_java>/java -jar machineagent.jar &
```

## Learn More

- [Setting Up an SSH Key for Controller Provisioning](#)
- [AppDynamics Support](#)

## Setting Up an SSH Key for Controller Provisioning

- [To set up SSH Key Pairs Using DSA](#)
- [To set up SSH Key Pairs Using RSA](#)

You can set up SSH (Secure Shell) with public/private key pairs so that you do not have to type the password each time that you ssh into the Controller machines. This allows scripts and automation processes to access the necessary systems easily. You can generate DSA or, if you want stronger encryption, RSA keys.

### To set up SSH Key Pairs Using DSA

1. Run the ssh command that sets up the key pair:

```
% ssh-keygen -t dsa
```

2 To the prompt:

```
Generating public/private dsa key pair.  
Enter file in which to save the key (~/.ssh/id_dsa):
```

just type RETURN.

3. To the prompt:

```
Enter passphrase (empty for no passphrase):
```

just type RETURN.

4. To the prompt:

```
Enter same passphrase again:
```

just type RETURN.

You should see the following information:

```
Your identification has been saved in \~/.ssh/id_dsa  
Your public key has been saved in \~/.ssh/id_dsa.pub  
The key fingerprint is: <Some really long string>
```

If SSH continues to prompt you for your password, verify your permissions in your remote .ssh directory. It should have only your own read/write/access permission (octal 700):

```
% chmod 700 \~/.ssh
```

5. Open the local ~/.ssh/id\_dsa.pub file and paste its contents into the ~/.ssh/authorized\_keys file on the remote host.

6. Update the permissions on the authorized\_keys file on the remote host as follows:

```
% chmod 600 \~/.ssh/authorized_keys
```

### To set up SSH Key Pairs Using RSA

Run the ssh command that sets up the key pair:

```
% ssh-keygen \-t rsa
```

The generated files will be named id\_rsa and id\_rsa.pub, instead of id\_dsa and id\_dsa.pub.

Otherwise, the remaining steps are identical to those beginning with step 2 in [To set up SSH Key Pairs Using DSA](#).

## Controller Data and Backups

This section discusses Controller data administration and backups.

### Controller Data Backup and Restore

- Controller Backups
  - Best Practices for Backup
  - Regular Hot Backups
  - Using a Hot Backup for the New Physical Server
  - System Settings and Tools
    - Why AppDynamics recommends binary backup
  - Using XtraBackup

- To backup Controller data using XtraBackup
- To restore Controller data using XtraBackup
- Backup and Restore Procedures when Running in High Availability (HA) Mode
  - To backup when using High Availability mode
  - To restore when using HA mode
- Backing Up Metadata Only
- Learn More

This topic describes the backup and restore procedure for the Controller database.

## Controller Backups

The AppDynamics Controller uses MySQL to store information about following components:

- Design of your applications (all meta-data about business transactions, tiers, policies etc.)
- History of the performance of your applications (metric data)
- Transaction Snapshot Data and Events
- History of incidents that occurred (both resolved and unresolved incidents are stored)

## Best Practices for Backup

 **Important:** AppDynamics strongly recommends that you **perform routine data backups** as a part of disaster recovery preparedness.

There are other situations when you should do a backup. Back up the Controller under the following conditions:

- During an upgrade
- When migrating your installation from one server to another
- If the Controller disk space is running low

## Regular Hot Backups

A "hot" backup of Controller data is recommended when you do not want to stop the Controller.

 **Important:** AppDynamics strongly recommends that you **perform nightly backups** of your data. If you cannot do nightly backups do backups at least once a week.

The AppDynamics Controller uses the MySQL default storage engine, InnoDB. Therefore, you cannot copy the data files directly while the Controller is running. Copying data files works only if the Controller is stopped. For details see [Start or Stop the Controller](#).

 **Warning:** Do not use tools like mysqldump because these tools convert the data into text format and are time-consuming, especially if you have gigabytes of data in your system.

## Using a Hot Backup for the New Physical Server

You can perform a hot or a cold backup, but it is important to get the backup of the data directory, located in <Controller\_Installation\_Directory>/db.

However, we recommend a cold backup of the data. Hot backup will not bring the Controller down for a long time, but you will still lose the data when you migrate. This is because hot backup will only have the data from the time that the hot backup started, and not when the backup was completed.

## System Settings and Tools

Ensure that <controller\_install\_dir>/db/bin is in your \$PATH variable and it should be ahead of any other path that has another MySQL. (Typically Linux has a standard instance of MySQL installed, which will conflict with the Controller's path.)

There are chances of data loss since your last backup and so more frequent backups are recommended.

It is recommended that you use only those tools that perform binary copies of the data.

### Why AppDynamics recommends binary backup

A binary data backup saves a copy of your directory data that you can use if the database files later become corrupted or deleted. For details see [Binary Backup](#).

- You can use the following backup tools for Linux:
  - mylvmbackup
  - Xtrabackup
  - INNODB Hot Backup
- You can use the following backup tool for Windows:
  - Zmanda Recovery Manager for MySQL

Additionally, you can also use the ZFS snapshot method to perform the backup for Controller database.

For details on the ZFS snapshot method, see: [Using ZFS methods for data backup](#).

## **Using XtraBackup**

### **To backup Controller data using XtraBackup**

You can use the Percona xtraBackup tool for the backup, available at: <http://www.percona.com>.

Use version 1.6.4-314 or greater. You can either use the Linux binary or the RPM install.

- Add the <xtrabackup\_dir>/bin directory path to the \$PATH variable.
- Use the following two commands for each backup (scroll right to see the whole command line):

```
innobackupex-1.6.4 --user=root--password=singcontroller --defaults-file=$INSTALL_DIR/db/db.cnf
<backup-dir> --no-lock

innobackupex-1.6.4 --user=root--password=singcontroller --use-memory=1GB
--defaults-file=/${INSTALL_DIR}/db/db.cnf --apply-log <backup-dir-from above>/<some-date-dir>
```

The first command creates a directory under your <Backup\_Directory> using the time-stamp.

Use the full directory name as the final argument to the second command.

You must run both commands for a successful backup.

### **To restore Controller data using XtraBackup**

1. Shut down the MySQL database.
2. Copy back the files in the data directory located at <Controller\_Installation\_Directory>/db.
3. Restart MySQL.

## **Backup and Restore Procedures when Running in High Availability (HA) Mode**

After you have configured the Controller in **high availability mode**, use the following data backups and restore procedures.

You can perform a hot backup on the secondary machine in an HA environment using mylvmbackup or Xtrabackup.

### **To backup when using High Availability mode**

DO NOT backup the Controller database. Use the following steps to ensure less impact on your running system.

1. Log into the Controller database on the secondary machine.
2. For every backup period (which should be nightly), shut down the Controller database on the secondary machine.
3. Use any incremental copy tool to copy the Controller database to a safe location.
4. Start the Controller database.

### **To restore when using HA mode**

Use the following steps when your primary MySQL server suffers a data loss, such as might happen after you switch your operations to

the secondary MySQL:

1. Shutdown the Controller database on primary machine, after it is repaired.
2. Copy the data from the secondary machine to the primary machine.
3. Start the primary machine. The primary machine should now start replicating any data that it has lost since the time the last backup performed on the secondary server.
4. Verify that the primary and secondary machines are in sync.
5. Switch back to using the primary machine as your Controller.

## Backing Up Metadata Only

You can perform a minimal backup that backs up only metadata. This is essentially a MYSQL dump. Click [MetadataBackupCommand.txt](#) to download the command script to back up the metadata.

## Learn More

- [Manage Controller High Availability](#)
- [App Agent for Java Configuration Properties](#)
- [App Agent for .NET Configuration Properties](#)

## Controller Database Scripts

### Database Scripts

The scripts to start or stop the Controller are located in the <Controller\_Installation\_Directory>/bin directory.

Action	Commands for Windows	Commands for Linux
Log into the Controller database	controller.bat login-db	./controller.sh login-db
Start the Controller database	controller.bat start-db	./controller.sh start-db
Stop Controller database	controller.bat stop-db	./controller.sh stop-db
Reset Controller database	controller.bat reset-db	./controller.sh reset-db

## Learn More

## Controller Data Storage

- [The Controller Data Directory](#)
- [Moving the Controller Data Directory](#)
  - To relocate the Controller data directory

This topic describes moving the Controller data storage directory.

### The Controller Data Directory

By default, the Controller's data directory is located at: <Controller\_Installation\_Directory>/db.

### Moving the Controller Data Directory

You may decide to move the data directory to a new location under following conditions:

- When you want to store the Controller data on a SAN in order to get higher I/O performance and redundancy.
- If there is not enough disk space available during Controller installation.

 **Note:** If you are using symlinks, you must create the symlink outside of the root Controller install directory and move the data directory to the new volume after you install the Controller.

 **Warning:** Do not mount a file system on <Controller\_Installation\_Directory>/db/data. During Controller upgrade, the installer moves the data directory to data\_orig. Upgrade will fail if the installer cannot complete this move.

### To relocate the Controller data directory

1. Stop the Controller and its database. See [Start or Stop the Controller](#).
2. Modify following properties in the <Controller\_Installation\_Directory>/db/db.cnf file to point to the new location of the data directory.

```
datadir  
tmpdir  
log  
slow_query_log_file
```

3. Copy (or move) the existing data directory <Controller\_Installation\_Directory>/db> to the new location.

For Linux:

```
>cd <Controller_Installation_Directory>/db/  
>cp data <new-location>
```

4. Start the Controller. See [Start or Stop the Controller](#).
5. Check the database.log and server.log for any database connection related errors.

## Controller Disk Space and the Database

- Disk Space Considerations
- Disk Space Warnings
- Managing Disk Space
- Automatic Shutdown When Disk Space is Low
  - To disable automatic shutdown for a running Controller
- Learn More

This topic discusses best practices for managing disk space for the MySQL database used by the Controller.

### Disk Space Considerations

If disk space runs out the Controller will not function and data may become corrupted.

### Disk Space Warnings

When disk space starts getting low, the AppDynamics UI displays a Controller Disk Space Low alert. The warning is triggered based on the Controller profile. See [Controller System Requirements](#).

AppDynamics also logs error in the server.log.

## Managing Disk Space

If the disk space is low, you need to reduce the size of the Controller database.

To manage how much disk space the Controller database uses, you change the amount of data retained in the Controller database. See [Database Size, Data Retention, and Metric Resolution](#).

### Automatic Shutdown When Disk Space is Low

Starting with Controller release 3.4.2, if the space on the disk the Controller is running on falls below 1 GB, the Controller automatically shuts down to avoid DB corruption.

#### To disable automatic shutdown for a running Controller

1. Shut down the Controller. See [Start or Stop the Controller](#).
2. Open the <Controller-Installation-Directory>/appserver/domains/domain1/config/domain.xml file for edit.
3. Add the following:

```
<jvm-options>-Dignore.disk.space=true</jvm-options>
```

4. Save the file.
5. Restart the Controller.

## Learn More

- [Database Size, Data Retention, and Metric Resolution](#)
- [Controller System Requirements](#)
- [Start or Stop the Controller](#)

## Database Size, Data Retention, and Metric Resolution

- [Information Stored by the Controller](#)
- [The Controller Database and Data Retention Settings](#)
  - To change the Controller data retention settings
  - Troubleshooting Controller Database Growth Issues
- [Default Data Retention Policy for Metrics](#)
  - Modifying Default Data Retention Policies
  - To change the data retention period for metrics
- [Learn More](#)

This topic explains the default data retention settings for data stored by the AppDynamics Controller, and how often AppDynamics resolves metric data.

### Information Stored by the Controller

The Controller holds information about following components:

- The design of your applications (all metadata about business transactions, tiers, policies, etc.)
- Several years of historical data about application performance (metric data)
- Several weeks of transaction snapshot data and events
- History of incidents that occurred (whether they were resolved or not)

The Controller uses MySQL to store this data.

### The Controller Database and Data Retention Settings

You can change the amount of data you retain in the Controller database by changing the retention period. You can also change the data retention policy for snapshots and hourly metric data.

You should consider changing the data retention settings when you see an increase in the size of the Controller database. See [Controller Disk Space and the Database](#).

You can also purge old data that is no longer needed by changing the retention settings.

### To change the Controller data retention settings

1. Log in to the Controller administration console using the root account password. See [Access the Administration Console](#).

2. Click **Controller Settings**.

3. Change the data retention period by configuring the following parameters:

- `events.retention.period`  
The default value is 336 hours, which is 2 weeks.
- `snapshots.retention.period`  
The default value is 336 hours, which is 2 weeks.

The size of the database should drop between 30-60 minutes after you make this change. You should see fewer `eventdata_` files. If you do not see this result [restart the Controller](#).

### Troubleshooting Controller Database Growth Issues

If changing the data retention settings does not restrict the database growth, send a directory listing of your data directory, in particular the `eventdata_` files, to the [AppDynamics Support Team](#).

Use the following command to get the directory listing of the data directory:

```
ls <install_dir>/db/data/controller eventdata_detail* | sort
```

There should be one file per hour of data.

### Default Data Retention Policy for Metrics

AppDynamics triggers data retention for metrics after the data has existed for 60 minutes.

Over time, AppDynamics resolves the data in the following manner:

- Initially, for up to 4 hours, all data is captured at full resolution, minute-by-minute, for each metric.
- After 4 hours, the Controller resolves the data every 10 minutes.
- After 48 hours, the Controller resolves the data every 60 minutes.
- Finally, daily averages are used as resolution.

AppDynamics resolves the metrics using averages on the data and discards the remaining metric data points.

The data retention policy aggregates the minute-by-minute data into ten-minute resolutions after 4 hours. After 48 hours, the ten-minute data is further aggregated to 60-minute resolutions.

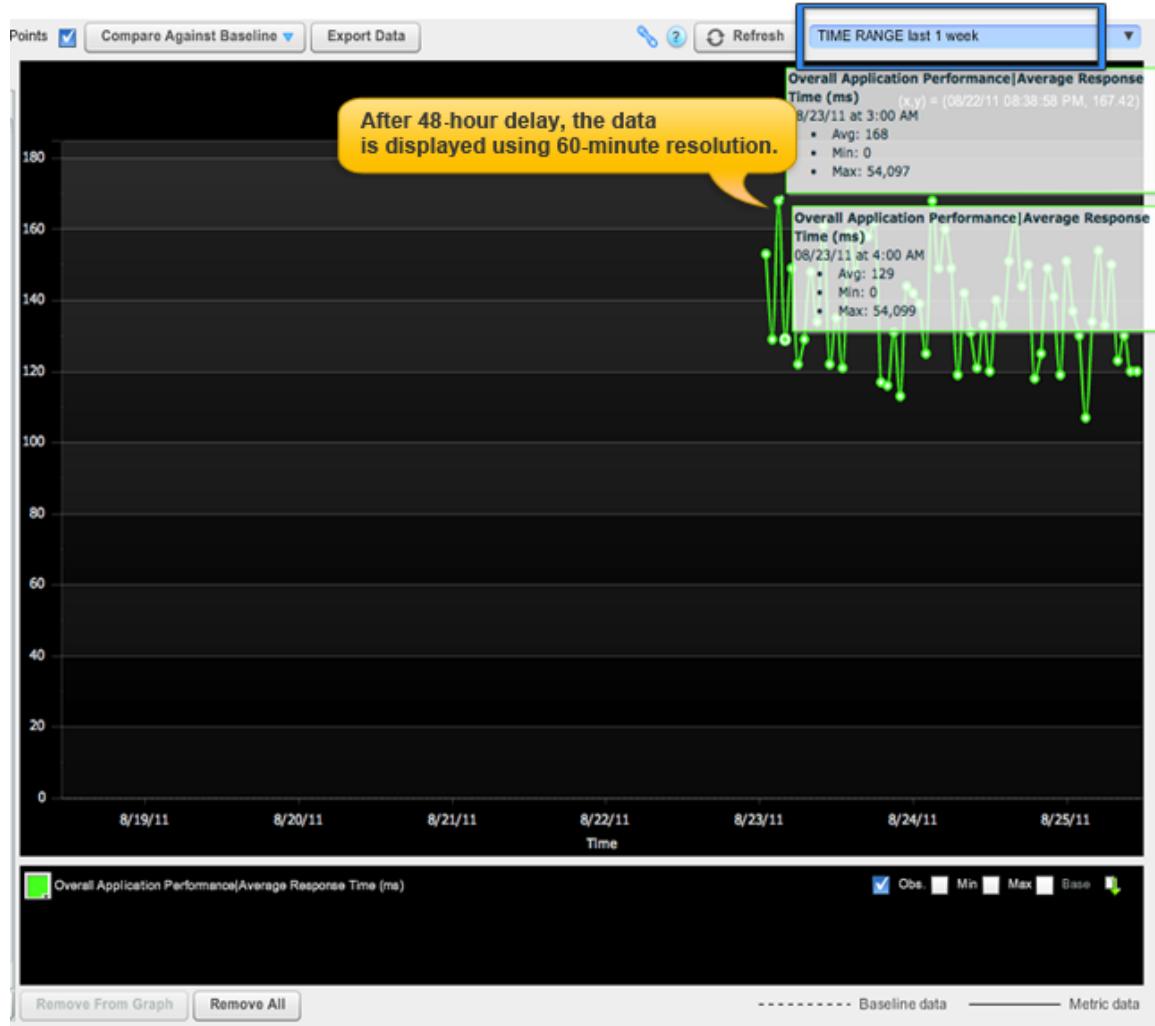
For example, if you select a four-hour time-range between 10.00 a.m. to 12.00 noon for a data which is 4 hours old, your data points will be determined by the following resolution: 10.00 a.m.-10.09 a.m., 11.00 a.m.-11.09 a.m., 12.00 noon - 12.09 p.m.

 **Important:** In the above example, if you select a custom time range of 10.00 a.m. to 12.01 p.m., it is likely that the data might not capture all the data points.

The following screenshot shows how the data retention policy determines the metric resolution. AppDynamics displays metric data after 4 hours, and any two data points are separated by a 10-minute time interval.



After 48 hours, AppDynamics displays the data using the 60-minute resolution.



To get the required visibility, select only those custom time ranges that follow the "data-resolution" rule.

## Modifying Default Data Retention Policies

You can modify the default data retention policies. However, due to performance reasons, AppDynamics strongly recommends that you exercise caution if you plan to increase the default limits.

The data retention policies are determined by three properties that are accessible from the Controller administration console.

Property name	About the property	Default	Allowed Values
metrics.min.retention.period <i>specified in hours</i>	This property determines the number of hours your minute-by-minute data will be retained.	4 hours	Allowed value is between 1 hour to 1 week (168 hours).
metrics.ten.min.retention.period <i>specified in hours</i>	This property determines the number of hours your ten-minute data will be retained.	48 hours	Allowed value is between 2 hours to 3 weeks (504 hours).
metrics.retention.period <i>specified in days</i>	This property determines the number of days the hour-by-hour metrics will be retained in the Controller database.	365 days	Allowed value is between 30 days to 2 years (730 days).

The value for metrics.min.retention.period should always be less than the value for metrics.ten.min.retention.period.

The value for metrics.ten.min.retention.period should always be less than the value for metrics.retention.period.

**⚠ Warning:** If you specify the value for minute data (metrics.min.retention.period) either less than 36 hours or greater than 36 hours (or vice versa) the Controller requires special repartitioning and therefore you will lose all the minute data for the first time after you save the settings.

### To change the data retention period for metrics

1. Log in to the Controller administration console.

```
http://<controller-installer-host>:<port>/controller/admin.html
```

Use the root account password to access the Admin console. The root password is "changeme" and is the same even if the Controller is installed in single or multi-tenant mode.

2. Click **Controller Settings**.

3. Change the data retention period by configuring the following parameters. Save each property individually.

- metrics.min.retention.period  
The default value is 4 hours.
- metrics.ten.min.retention.period  
The default value is 48 hours.
- metrics.retention.period  
The default value is 365 days.

After you modify these values, the time range selector in the UI that displays "resolution X minute" should change.

For example, if the metrics.min.retention.period property is changed to 3, in the AppDynamics UI all time ranges <= 3 hours should have resolution of 1 minute. If the metrics.ten.min.retention property is changed to 168, all time ranges <= 168 hours (1 week) and greater or equal to min.retention should have resolution of 10 minutes.

### Learn More

- [Controller Disk Space and the Database](#)
- [Controller Database Scripts](#)
- [Infrastructure Metrics](#)

## Controller Info Configuration for Agents

- [Learn More](#)

The Controller gets information from app agents and machine agents about how the agent connects to the Controller from the agent's controller-info.xml file.

For app agents, the controller-info.xml file is in the <Agent\_Installation\_Directory>/conf directory.

For machine agents, the controller-info.xml file is in the <Machine\_Agent\_Installation\_Directory>/conf directory.

The properties set in the controller-info.xml file include:

- controller-host
- controller-port
- controller-ssl-enabled
- application-name
- tier-name
- node-name
- agent-runtime-dir
- enable-orchestration
- account-name

- account-access-key
- force-agent-registration

System properties in the app server startup script override the settings in controller-info-xml.

For information about the controller-info settings, as well as their corresponding system, properties, see [App Agent for Java Configuration Properties](#), [App Agent for .NET Configuration Properties](#) and [Machine Agent Configuration Properties](#).

## Learn More

- [App Agent for Java Configuration Properties](#)
- [App Agent for .NET Configuration Properties](#)
- [Machine Agent Configuration Properties](#)

## Configure Controller VIP

If you want to use a virtual IP (VIP) for the Controller, perform the following edits in the <Controller-Installation-Directory>/appserver/domains/domain1/config/domain.xml file:

1. Set the appdynamics.com.hostname element to the internal IP address that you used when you installed the Controller.
2. Set the appdynamics.com.server.hostname to the virtual IP address.