



APPDYNAMICS

Troubleshooting

AppDynamics Pro Documentation

Version 3.8.x

1. Rapid Troubleshooting	3
1.1 Troubleshoot Slow Response Times	4
1.2 Troubleshoot Expensive Methods and SQL Statements	7
1.3 Troubleshoot Node Problems	11
1.4 Troubleshoot Errors	13
1.5 Troubleshoot a Problem that Happened in the Past	16
1.6 Diagnostic Sessions	21
1.7 Call Graphs	23
1.7.1 Configure Call Graphs	29
1.8 Analyze	32
1.8.1 Scalability Analysis	32
1.8.2 Correlation Analysis	34
1.8.3 Compare Releases	35
2. Troubleshoot Health Rule Violations	36
3. Troubleshoot Java Application Problems	42
3.1 Troubleshoot Slow Response Times for Java	42
3.2 Configure Diagnostic Sessions For Asynchronous Activity	59
3.3 Troubleshoot Java Memory Issues	60
3.3.1 Troubleshoot Java Memory Leaks	60
3.3.2 Troubleshoot Java Memory Thrash	68
3.4 Detect Code Deadlocks for Java	75
4. Troubleshooting Tutorials for Java	77
4.1 Tutorial for Java - Business Transaction Health Drilldown	77
4.2 Tutorial for Java - Exceptions	77
4.3 Tutorial for Java - Slow Transactions	83
4.4 Tutorial for Java - Troubleshooting using Events	86
5. Troubleshoot .NET Application Problems	93
5.1 Troubleshoot Slow Response Times for .NET	93
6. Troubleshoot PHP Application Problems	104
6.1 Troubleshoot Slow Response Times for PHP	106
6.2 Troubleshoot Errors for PHP	108
7. Troubleshoot Node.js Application Problems	112
7.1 Troubleshoot Slow Response Times for Node.js	112
8. Troubleshoot Mobile Applications	114
8.1 Troubleshoot Slow Network Requests from Mobile Applications	114
8.2 Troubleshoot Mobile Application Crashes	115

Rapid Troubleshooting

- [Troubleshooting Scenarios](#)
- [How AppDynamics Indicates Problems](#)
- [How to Start Troubleshooting](#)

Troubleshooting Scenarios

For common troubleshooting scenarios see:

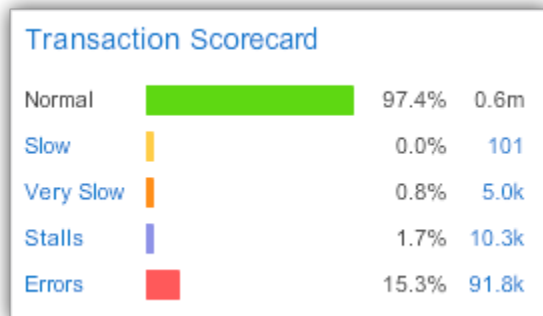
- [Troubleshoot Slow Response Times](#)
- [Troubleshoot Errors](#)
- [Troubleshoot Health Rule Violations](#)
- [Troubleshoot Expensive Methods and SQL Statements](#)
- [Troubleshoot Mobile Applications](#)

Troubleshoot per Platform

- [Troubleshoot Java Application Problems](#)
- [Troubleshoot .NET Application Problems](#)
- [Troubleshoot PHP Application Problems](#)
- [Troubleshoot Node.js Application Problems](#)

How AppDynamics Indicates Problems

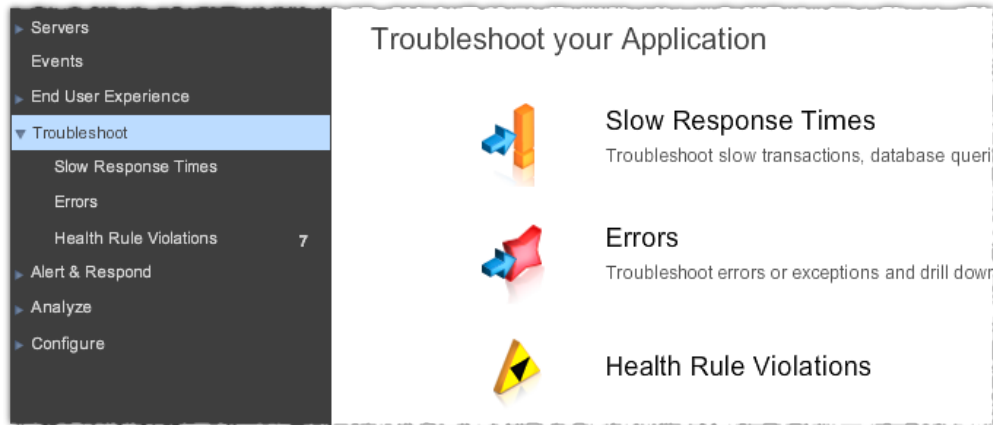
The [dashboards](#) show you when problems occur and you need to take action. For example the [Transaction Scorecard](#) monitors business transactions and categorizes their performance according to thresholds.



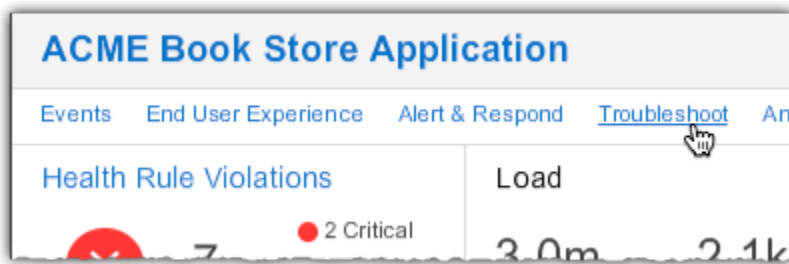
How to Start Troubleshooting

When AppDynamics indicates a problem you can easily go right into troubleshooting mode.

- From the left navigation menu click Troubleshoot.



- From the All Applications dashboard click the Troubleshoot link.



Troubleshoot Slow Response Times

- How You Know When Response Times are Slow
- Basic Troubleshooting Techniques
 - Slow and Stalled Transactions
 - To troubleshoot slow and stalled transactions
 - Slow Database and Remote Service Calls
 - To troubleshoot slow database and remote service calls
- Learn More

Troubleshoot per Platform

[Troubleshoot Slow Response Times for Java](#)
[Troubleshoot Slow Response Times for .NET](#)
[Troubleshoot Slow Response Times for PHP](#)
[Troubleshoot Slow Response Times for Node.js](#)
[Troubleshoot Slow Network Requests from Mobile Applications](#)

How You Know When Response Times are Slow

There are many ways you can learn that your application's response time is slow:

- You received an email or SMS alert from AppDynamics (see [Alert and Respond](#)). The alert

provides details about the problem that triggered the alert.

- Someone reported a problem such as "it's taking a long time to check out" or "the app timed out when I tried to add an item to the cart."
- A custom dashboard shows a problem.
- An AppDynamics dashboard shows a problem. In AppDynamics, look at:
 - Business Transaction health metrics in the Transaction Scorecard pane. The bar charts show slow or stalled transactions (as based on health rules) as compared to normal.
 - The Response Time graph. If you see spikes in the graph, the problem is slow response time.
 - Events in the Events List. Scroll the list and look for red or yellow icons related to performance health rules based on response time.
 - Traffic flow lines in a flow map. If you see yellow or red flow lines the problem is likely to be slow response time.
 - Business Transaction Health status in the Business Transaction Health pane. If you see yellow or red bars, there are health rule violations that may affect response time.
 - Tier icons in the flow map. If the icon is yellow or red there is a problem with one or more nodes that may affect response time.
 - Health rule status in the Server Health pane. If you see red or yellow, the problem reflects a server health rule violation that may affect response time.

Basic Troubleshooting Techniques

At any time you can click **Troubleshoot -> Slow Response Times** and the Slow Response Times window opens showing two tabs. You can drill down into transaction issues in the [Slow Transactions tab](#), and into database or remote services issues in the [Slowest DB & Remote Services tab](#).

Slow and Stalled Transactions

There are many reasons why a business transaction may be slow or stalled. The Slow Response Times tab helps you find the root cause whether that be resource or thread contention, deadlock, race condition, or something else.

By default AppDynamics considers a slow transaction one that lasts longer than 3 times the standard deviation for the last two hours and a very slow transaction 4 times the baseline for the last two hours.

By default AppDynamics considers a transaction that lasts longer than 45 seconds (4500 milliseconds) to be stalled.

You can configure these thresholds to better match your environment. See [Thresholds](#) and [Configure Thresholds](#).

To troubleshoot slow and stalled transactions

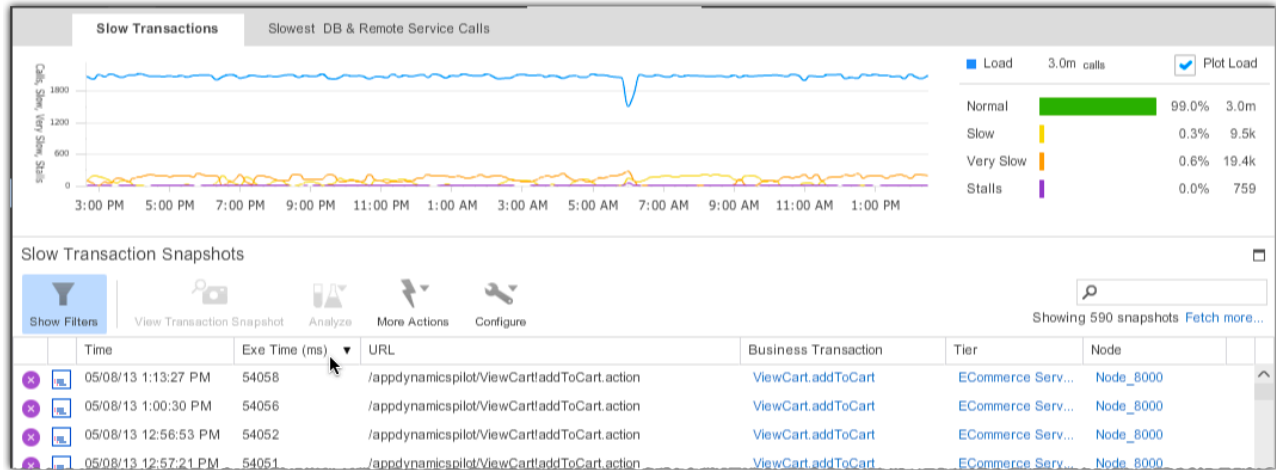
1. Click **Troubleshoot -> Slow Response Times**.

You can also access this information from tabs in the various dashboards.

2. Click the **Slow Transactions** tab if it is not selected.

- In the upper pane AppDynamics displays a graph of the slow, very slow, and stalled transactions for the time period specified in the Time Range drop-down menu. If the load is not displayed, you can click the Plot Load checkbox at the upper right to see the load.
- In the lower pane AppDynamics displays the transaction snapshots for slow, very slow, and stalled transactions.

3. In the lower pane, click the Exe Time column to sort the transactions from slowest to fastest.



To drill down to the root cause of the slow or stalled transaction, select a snapshot from the list and click **View Transaction Snapshot**. See [Transaction Snapshots](#).

Slow Database and Remote Service Calls

Although AppDynamics does not instrument database and remote service servers directly, it collects metrics about calls to these backends from the instrumented app servers. This allows you to drill down to the root cause of slow database and remote service calls.

To troubleshoot slow database and remote service calls

1. Click **Troubleshoot -> Slow Response Times**.

You can also access this information from tabs in the various dashboards.

2. Click the **Slowest DB & Remote Service Calls** tab if it is not selected.

- In the Call Type panel, you can select the type of call for which you want to see information, or select All Calls.
- The Call panel displays the average time per call, number of calls, and maximum execution time (Max Time) for the calls with the longest execution time.
- In the lower panel, you can see details or correlated snapshots for the selected call.

Slow Transactions

Slowest DB & Remote Service Calls

Call Type

All Calls

JDBC

JMS

These are the calls with largest observed individual execution time (Max Time) during the specified time range.

Call	Avg. Time per Call (ms)	Number of Ca	Max Time (ms)	Snapshots
SELECT COUNT(1) COUNT FROM ITEM IT1, ITEM IT2	7521.5	1501	31124	View snapshots
ORDERSERVICE.CREATEORDER	1596.8	5196	20013	View snapshots
INSERT INTO ORDERREQUEST (ITEM_ID, NOTES) VALUE:	950.4	5759	10002	View snapshots
GET POOLED CONNECTION FROM DATASOURCE	2380.7	234	9883	View snapshots
SELECT ITEM.ID, NOW(), SLEEP(1/3), NOW() FROM ITEM WI	333.9	5759	339	View snapshots
ORDERQUEUE	110	1	110	No snapshots
DB TRANSACTION COMMIT	11.8	1987	79	View snapshots
DB TRANSACTION COMMIT	11.8	76	26	View snapshots
INSERT INTO CART (ITEM_ID, USER_ID) VALUES (?, ?)	15.9	28	25	No snapshots

Call Details

Correlated Snapshots

SELECT COUNT(1) COUNT FROM ITEM IT1, ITEM IT2

3. Sort by Average Time per Call to display the slowest calls at the top of the list.

4. To see transaction snapshots for the business transaction that is correlated with a slow call, you can:

- Click the **View Snapshots** link in the right column to display correlated snapshots in a new window.
- Select the call and click the **Correlated Snapshots** tab in the lower panel to display correlated snapshots at the bottom of the screen.

Call Details		Correlated Snapshots					
		View Transaction Snapshot					
		Time	Exe Time (r	URL	Business Transaction	Tier	Node
		03/27/14 4:49:37 PM	9868	/appdynamicsp	Fetch Catalog	E-Commerce	E-Commerce-N...
		03/27/14 4:50:11 PM	10246	/appdynamicsp	Fetch Catalog	E-Commerce	E-Commerce-N...
		03/27/14 4:50:13 PM	10440	/appdynamicsp	Fetch Catalog	E-Commerce	E-Commerce-N...
		03/27/14 4:50:59 PM	10518	/appdynamicsp	Fetch Catalog	E-Commerce	E-Commerce-N...
		03/27/14 4:51:22 PM	11392	/appdynamicsp	Fetch Catalog	E-Commerce	E-Commerce-N...

5. Click the Exe Time column to sort the transactions from slowest to fastest.

To drill down to the root cause of the slow call, select a snapshot from the list and click **View Transaction Snapshot**. See [Transaction Snapshots](#).

Learn More

- [Transaction Snapshots](#)
- [Configure Thresholds](#)

Troubleshoot Expensive Methods and SQL Statements

Expensive Method Invocations and SQL Statements

If a part of your application is showing performance degradation and you're not sure where to start looking for the root cause,

one approach is to search and compare multiple snapshots together. You can do this to find the most expensive method invocations and SQL statements.

To troubleshoot by comparing multiple snapshots

1. Select the application dashboard and click the Transaction Snapshot tab.
2. Click the All Snapshots tab.

	Time	Exe Time (ms)	URL
✖	02/01/13 2:04:13 PM	8	/consumer/logout.aspx
⚠	02/01/13 2:03:43 PM	2780	/consumer/misc.aspx
⚠	02/01/13 2:03:38 PM	2480	/consumer/misc.aspx
✔	02/01/13 2:03:15 PM	116	/consumer/UpdateOrders.aspx
✔	02/01/13 2:02:46 PM	224	/consumer/misc.aspx
⚠	02/01/13 2:02:41 PM	2476	/consumer/misc.aspx
⚠	02/01/13 2:02:39 PM	2490	/consumer/misc.aspx
✔	02/01/13 2:02:38 PM	221	/consumer/misc.aspx
⚠	02/01/13 2:02:36 PM	2021	/consumer/misc.aspx
✔	02/01/13 2:02:21 PM	96	/consumer/ProductFetch.aspx
✔	02/01/13 2:02:12 PM	2	/consumer/login.aspx
✖	02/01/13 2:02:11 PM	1	/consumer/login.aspx
⚠	02/01/13 2:01:41 PM	2430	/consumer/misc.aspx
⚠	02/01/13 2:01:38 PM	2443	/consumer/misc.aspx
⚠	02/01/13 2:01:36 PM	2588	/consumer/misc.aspx
✔	02/01/13 2:00:38 PM	224	/consumer/misc.aspx
✔	02/01/13 2:00:32 PM	226	/consumer/misc.aspx
✔	02/01/13 2:00:25 PM	122	/consumer/SaveOrder.aspx
✖	02/01/13 1:59:12 PM	4	/consumer/logout.aspx
✔	02/01/13 1:58:38 PM	1994	/consumer/misc.aspx
✔	02/01/13 1:58:12 PM	7	/consumer/logout.aspx
✔	02/01/13 1:57:41 PM	229	/consumer/misc.aspx

3. Sort the snapshots in a way that makes sense to you, such as by tier and by business transaction.
4. Select a few snapshots. You can compare up to 30 snapshots at a time.
5. Click **Analyze -> Identify the most expensive calls/SQL statements in a group of Snapshots**.

demo1.appdynamics.com/controller/#location=APP_DASHBOARD&timeRange=last_15_minutes.BEFORE_NOW.-1.1359753853688.15&application=14

AppDynamics Pro

NWTraders

Business Transactions

Servers

App Servers

Consumer Web

Fedex Service

Machine Agent

Order Service

Order Service-AMAZONA

Order Service-AMAZONA

Product

Queue Service

Databases

Remote Services

IIS App Pools

Events

End User Experience

Troubleshoot

Slow Response Times

Errors

Policy Violations

Analyze

Configure

Instrumentation

Policies

Alerts

Slow Transaction Thresholds

Baselines

Dashboard

Top Business Transactions

Transaction Snapshots

Transaction Analysis

All Snapshots

Slow and Error Transactions

Diagnostic Sessions

Periodic Collection

Show Filters

View Transaction Snapshot

Analyze

More Actions

Configure

Compare Snapshots

Identify the most expensive calls / SQL statements in a group of Snapshots

	Time	Exe Time	
	02/01/13 1:56:42 PM	2501	
	02/01/13 1:50:32 PM	246	/consumer/misc.aspx
	02/01/13 1:50:35 PM	249	/consumer/misc.aspx
	02/01/13 1:50:35 PM	2426	/consumer/misc.aspx
	02/01/13 1:50:38 PM	2419	/consumer/misc.aspx
	02/01/13 1:50:41 PM	2447	/consumer/misc.aspx
	02/01/13 1:50:43 PM	2457	/consumer/misc.aspx
	02/01/13 1:57:41 PM	229	/consumer/misc.aspx
	02/01/13 2:02:46 PM	224	/consumer/misc.aspx
	02/01/13 1:56:40 PM	2028	/consumer/misc.aspx
	02/01/13 1:52:32 PM	3863	/consumer/misc.aspx
	02/01/13 1:52:36 PM	222	/consumer/misc.aspx
	02/01/13 1:52:39 PM	2245	/consumer/misc.aspx
	02/01/13 1:52:42 PM	239	/consumer/misc.aspx
	02/01/13 2:03:43 PM	2780	/consumer/misc.aspx
	02/01/13 1:54:36 PM	226	/consumer/misc.aspx
	02/01/13 1:54:37 PM	2981	/consumer/misc.aspx
	02/01/13 1:54:40 PM	2481	/consumer/misc.aspx
	02/01/13 1:54:43 PM	2464	/consumer/misc.aspx
	02/01/13 2:03:38 PM	2480	/consumer/misc.aspx
	02/01/13 1:55:32 PM	268	/consumer/misc.aspx
	02/01/13 2:02:38 PM	221	/consumer/misc.aspx
	02/01/13 1:55:35 PM	2059	/consumer/misc.aspx

Depending on how many snapshots you chose, it may take more time to process the results.

Expensive Methods and SQL Statements

Most Expensive Methods / SQL Statements

Displaying up to the top 25 most expensive methods and SQL statements from the following requests:

	Time	Exe Time (ms)	URL	Business Transaction	Tier	Node
1	02/01/13 1:52:32 PM	3863	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0
2	02/01/13 1:54:37 PM	2981	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0
3	02/01/13 2:03:43 PM	2780	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0
4	02/01/13 1:56:42 PM	2501	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0
5	02/01/13 1:54:40 PM	2481	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0
6	02/01/13 2:03:38 PM	2480	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0
7	02/01/13 1:54:43 PM	2464	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0
8	02/01/13 1:50:43 PM	2457	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0
9	02/01/13 1:50:41 PM	2447	/consumer/misc.aspx	Checkout	Consumer Web	Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0

Expensive Methods (137)

Expensive SQL (21)

The following methods were the most expensive methods invoked:

Class and Method	Total Exe. Time(ms)	Avg. Exe. Time(ms)	Call Count
System.Data.SqlClient.SqlCommand:ExecuteNonQuery	32343	1540.1	21
System.ServiceModel.Dispatcher.RequestChannelBinder:Request	773	36.8	21
:SNIReadSync	26	13.0	2
System.Web.Services.Protocols.SoapHttpClientProtocol:Invoke	81	11.6	7
System.ServiceModel.Channels.RequestChannel:Request	244	8.7	28
System.ServiceModel.Dispatcher.RequestChannelBinder:Request	174	8.3	21
System.ServiceModel.Channels.RequestChannel:Request	189	6.8	28
Http Handler - System.Web.Script.Services.ScriptHandlerFactory+HandlerWrapper:ProcessRequest	47	6.7	7
System.ServiceModel.Channels.SecurityChannelFactory`1+SecurityRequestChannel:Request	93	6.6	14
System.ServiceModel.Channels.SecurityChannelFactory`1+SecurityRequestChannel:Request	90	6.4	14
WCF Service - System.ServiceModel.Dispatcher.SyncMethodInvoker:Invoke	74	5.3	14
System.Messaging.MessageQueue:SendInternal	16	2.3	7
WCF Service - System.ServiceModel.Dispatcher.SyncMethodInvoker:Invoke	12	1.7	7
Http Handler - ASP.mscx:ProcessRequest	2	0.1	20
WCF Service - System.ServiceModel.Dispatcher.SyncMethodInvoker:Invoke	1	0.0	21
System.IdentityModel.Selectors.SecurityTokenProvider:GetToken	0	0.0	14
misc:Page_Load	0	0.0	20
System.ServiceModel.Security.IssuanceTokenProviderBase`1:DoNegotiation	0	0.0	14
System.ServiceModel.Security.SecuritySessionClientSettings`1+ClientSecuritySessionChannel:OnOpen	0	0.0	14
System.Reflection.RuntimeMethodInfo:Invoke	0	0.0	28
System.Data.SqlClient.TdsParserStateObject:ReadSni	0	0.0	2
System.ServiceModel.Channels.ServiceChannelProxy:ExecuteMessage	0	0.0	14

6. The Expensive Methods and SQL Statements window shows the most expensive method invocations sorted by total time. You can also sort by average execution time or call count.

7. You can click on a snapshot to see its details.

8. Select two snapshots for the same business transaction and select **Analyze -> Compare Snapshots** to directly compare them.

Snapshot Comparison

Comparing the following two snapshots:

Snapshot 1 - Taken at 3:20:34 PM on 02/01/13

Execution time: 284 ms

Occurred on transaction: Checkout

Occurred on node: Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0

View

Snapshot 2 - Taken at 3:20:29 PM on 02/01/13

Execution time: 230 ms

Occurred on transaction: Checkout

Occurred on node: Consumer Web-AMAZONA-7UI9E1N-NWTraders-0-1-0

View

Methods in Both Snapshots

Methods in One Snapshot

Methods Invoked in Both Snapshots

The following methods were invoked in both snapshots. A positive value listed in the "Overall Change" column indicates that more time was spent in that method in Snapshot 2 than in Snapshot 1. Adjusting the slider on the right will remove methods whose time spent changed between snapshots was less than the slider value.

1 ms

52 ms

Class and Method	Time in S1 (ms)	Time in S2 (ms)	Change (ms)	Call Count
System.ServiceModel.Channels.SecurityChannelFactory`1+SecurityRequestChannel:Request	15	23	8	2
System.ServiceModel.Dispatcher.RequestChannelBinder:Request	114	115	1	3
WCF Service - System.ServiceModel.Dispatcher.SyncMethodInvoker:Invoke	1	2	1	1
WCF Service - System.ServiceModel.Dispatcher.SyncMethodInvoker:Invoke	0	1	1	3
System.Messaging.MessageQueue:SendInternal	4	6	2	1
System.ServiceModel.Channels.RequestChannel:Request	29	32	3	4
System.ServiceModel.Channels.RequestChannel:Request	43	51	8	4
System.Data.SqlClient.SqlCommand:ExecuteNonQuery	120	172	52	1

Learn More

- [Transaction Snapshots](#)

Troubleshoot Node Problems

- [To Access the Node Problem Viewer](#)
- [Node Problem Viewer](#)
- [Filter Options for Node Problem Analysis](#)
 - [Baseline to Use](#)
 - [Data to Analyze](#)
 - [Analysis Type](#)
- [Learn More](#)

AppDynamics categorizes the ten items that deviate the most from the baseline performance as node problems.

You can analyze node problems from the Node Problems viewer.

The Machine Agent must be installed on the machine hosting the node that you are troubleshooting.

To Access the Node Problem Viewer

To access the Node Problems viewer do one of the following:

- In the Node dashboard, from the Actions drop-down menu click **Analyze Node Problems**.

or

- In the left navigation panel of the Snapshot viewer, click **NODE PROBLEMS**.

Node Problem Viewer

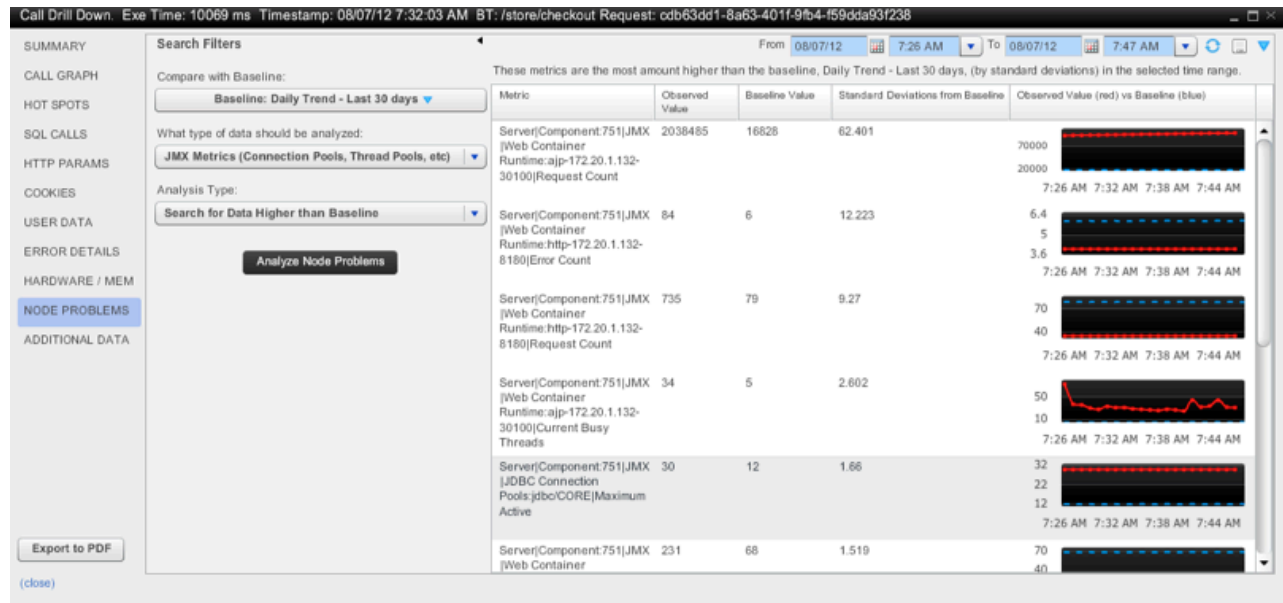
The right panel of the Node Problem viewer displays the metrics that deviate the most from their baselines for the specified time range.

The left panel of the Node Problem viewer lets you filter the types of data reported as node problems.

If accessed from the Snapshot viewer, the Node Problem viewer displays node problems for the time range of the snapshot. If accessed from the Node dashboard, it uses the time range set in the Node dashboard. You can edit the time range in the Node Problem viewer and then apply the new time range. You can also define and save a custom time range.

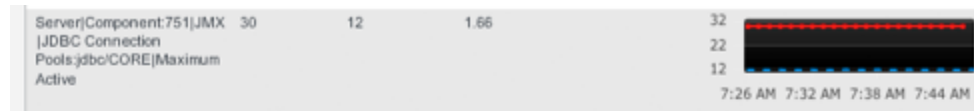
The selection of metrics displayed depends on the configured filter options.

The following Node Problem viewer shows a transaction that took over 10 seconds (10069 ms) to execute.



It shows that the request count was 62 standard deviations from its normal baseline and that the DB connection pool was 1.6 standard deviations above its normal baseline. It maxed out at 30 concurrent connections.

The **Hot Spots** tab for this snapshot shows the details of the connection pool latency.



Call Drill Down: Exe Time: 10069 ms Timestamp: 08/07/12 7:32:03 AM BT: /store/checkout Request: cdb63dd1-8a63-401f-9fb4-f59dda93f238

SUMMARY
CALL GRAPH
HOT SPOTS
SQL CALLS
HTTP PARAMS
COOKIES
USER DATA
ERROR DETAILS
HARDWARE / MEM
NODE PROBLEMS

This screen displays all of the method calls in the Call Graph sorted by time

Name	Method Time (ms)	External Calls
atg.adapter.gsa.GSATransaction:getConnection:732	185 ms (self) 1.8 %	JDBC
atg.adapter.gsa.GSATransaction:getConnection:732	118 ms (self) 1.2 %	JDBC
atg.adapter.gsa.GSATransaction:getConnection:732	39 ms (self) 0.4 %	JDBC
atg.adapter.gsa.GSATransaction:getConnection:732	60 ms (self) 0.6 %	JDBC
atg.adapter.gsa.GSATransaction:getConnection:732	117 ms (self) 1.2 %	JDBC
atg.adapter.gsa.GSATransaction:getConnection:732	45 ms (self) 0.4 %	JDBC
atg.adapter.gsa.GSATransaction:getConnection:732	144 ms (self) 1.4 %	JDBC
atg.adapter.gsa.GSATransaction:getConnection:732	66 ms (self) 0.7 %	JDBC
atg.adapter.gsa.GSATransaction:getConnection:732	240 ms (self) 2.4 %	JDBC

In this case, the Node Problems viewer reveals how excessive web requests saturated the DB connection pool.

Filter Options for Node Problem Analysis

You can filter the following options in the Filter Options panel of the Node Problem viewer:

- Baseline to Use
- Data to Analyze
- Analysis Type

Baseline to Use

Specify which baseline to use to define node problems:

- No baseline
- All data - Last 15 days
- Daily Trend - Last 30 days (default)
- Weekly Trend - Last 6 months
- Monthly Trend - Last year

For information about baselines, see [Behavior Learning and Anomaly Detection](#).

Data to Analyze

Select the type of data to analyze from the drop-down menu. AppDynamics analyzes the ten items that deviate the most from the baseline performance for the specified type.

Analysis Type

Specify whether to display problems with values that are:

- higher than baseline
- lower than the baseline
- higher and lower than baseline

For example, if you are only interested in CPU that is too high, set the Analysis Type to Higher for Hardware Resources. On the other hand, if you want to monitor the load on your machine continuously because low CPU usage would also be a problem, set the Analysis Type to Higher and Lower.

Learn More

- [Behavior Learning and Anomaly Detection](#)

Troubleshoot Errors

- [Error Transactions and Exceptions](#)
 - [To Troubleshoot Error Transactions](#)
 - [To Troubleshoot Exceptions](#)
- [Learn More](#)

Identifying and troubleshooting errors in your application.

Error Transactions and Exceptions

Metrics on errors are collected in addition to normal business transaction metrics.

An error is defined as one of the following:

- An *unhandled* exception in the context of a business transaction.
For example:
 - Business transaction entry point -> some code -> some exception thrown -> business

transaction finished -> exception caught. In this case AppDynamics reports the exception.

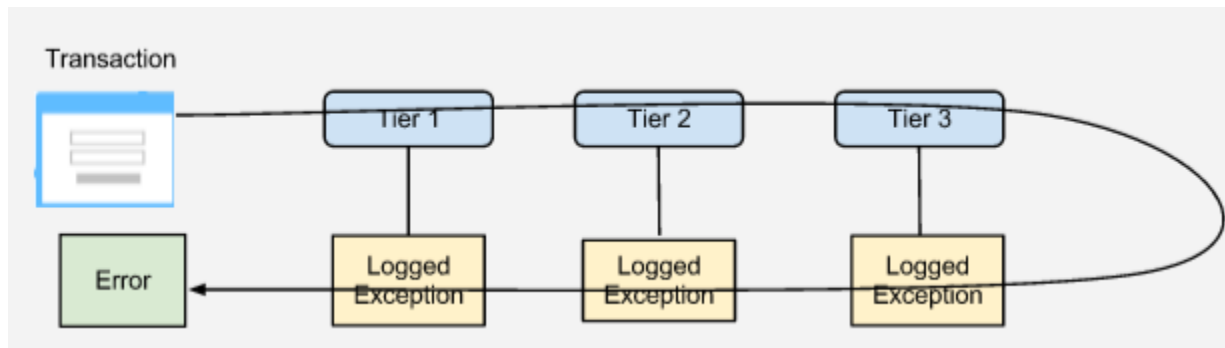
- Business transaction entry point -> some code -> some exception thrown -> exception caught inside a business transaction -> business transaction finished. In this case AppDynamics does not report the exception.

i If a business transaction experiences an error, it is counted as an error transaction and not as a slow, very slow or stalled transaction, even if the transaction was also slow or stalled.

- Any error that is logged with a severity of Error or Fatal using Log4j or java.util.logging (in Java), and Log4Net/NLog (in .NET) even if occurs outside the context of a business transaction. Depending on the signature of the method being called, you might not see an exception stack trace:
 - `logger.log(Level.ERROR, String msg, Throwable e)` - stack trace available
 - `logger.log(Level.ERROR, String msg)` – no stack trace here
- Any exception that occurs during an exit call, for example while calling SQL, a web service, or a message queue server.

An application server exception is a code-logged message outside the context of a business transaction.

There is not a one-to-one correspondence between the number of errors and the number of exceptions. For example, a business transaction may experience a single code 500 error in which several exceptions were logged as the transaction passed through multiple tiers.



You can configure the types of errors that AppDynamics detects as well as the types of exceptions to ignore. See [Configure Error Detection](#).

See the Supported Environments and Versions, such as [Supported Environments and Versions for Java](#) document for your app agent to determine if the loggers you use are recognized by default by AppDynamics. If you expect to see errors from a custom logger you first need to configure AppDynamics to recognize the logs. See [Custom Logger Definitions](#).

To Troubleshoot Error Transactions

1. Click **Troubleshoot -> Errors** in the left navigation panel.

The error viewer opens.

2. Click the **Error Transactions** tab if it is not already selected.

3. From the time range drop-down menu select the time range for which you want to view information about error transactions.

A graph of the error transactions displays at the top of the viewer. You can get an exact count of the errors per minute at a particular point in time by hovering with your pointing device on the line in the graph.

To the right of the graph is a summary of the load and the error transactions.

Check the Plot check box if you want the graph at the top of the viewer to show the load over the selected time period. Clear this check box is clear, if you want the graph to show only the error transactions .

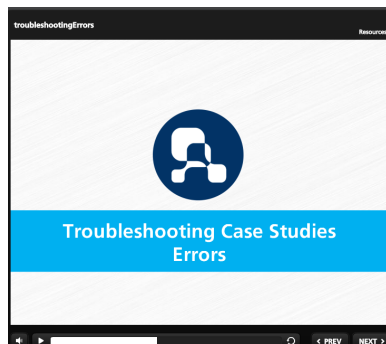
4. The error transaction snapshots are listed in the lower part of the viewer. To filter this list click **Show Filters** and select the filter criteria.

5. To examine the root cause of an error, select the snapshot from the list and click **View Transaction Snapshot**. See [Transaction Snapshots](#) for information about examining snapshots.

6. To identify the most expensive calls or queries, select a snapshot from the list and click **Analyze** and then click **Identify the most expensive calls / SQL statements in a group of snapshots**.

The Most Expensive Methods / SQL Statements viewer opens.

7. In the lower panel click the **Expensive Methods** tab to view the methods with their total and average execution times and call counts. Click the **Expensive SQL** tab to view the queries with their counts and execution times.



This two-minute interactive video traces the typical steps of identifying the cause of an error in your application.

To Troubleshoot Exceptions

1. Click **Troubleshoot -> Errors** in the left navigation panel.
2. Click the **Exceptions** tab if it is not already selected.

The total exception count, HTTP Error Codes and Error Page Redirects for the selected time

range are reported in the upper panel. You can get an exact count of the errors per minute at a particular point in time by hovering with your pointing device on the line in the graphs.

The exceptions list is displayed in the lower panel.

- To filter the exception list, enter the filter term in the search text box on the upper right. For example, to see only HTTP errors - and to see the breakdown of HTTP errors by code type - type HTTP in the search box. The list of HTTP errors by code is displayed.
- To see only errors with performance data, clear the Show Exceptions with 0 count checkbox.

3. To view details of a particular exception, select the exception the list in the lower panel and click **View Details**.

The exception detail window displays.

4. To view transaction snapshots for an exception:


- a. In the exception detail window, click the **Occurrences of this Exception** tab.
- b. Select a snapshot from the list.
- c. Click **View Details**.
- d. in the snapshot flow map that displays, click **Drill Down**. See [Transaction Snapshots](#).

5. To view a stack trace for an exception:

- a. In the exception detail window, click the **Stack Traces for this Exception** tab.
- b. Click an exception in the left panel.

The right panel displays the stack trace for the selected exception.

Learn More

- [Troubleshoot Errors](#) 
- [Configure Error Detection](#)
- [Transaction Snapshots](#)

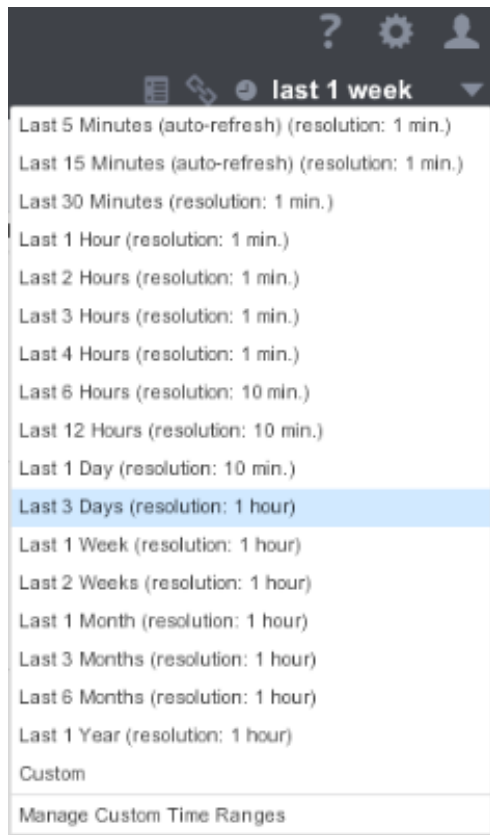
Troubleshoot a Problem that Happened in the Past

- [Set the Time Range](#)
- [Drill Down to the Problem](#)
- [Changing Other Time Windows](#)
- [Baselines](#)
- [Learn More](#)

It's not uncommon to want to investigate an issue that occurred in the past. For example, over the weekend there was a spike in stalled requests. The issue resolved, so there was no emergency, but, come Monday morning, you want to look back at what happened, and what was going on at the time in general in your system.

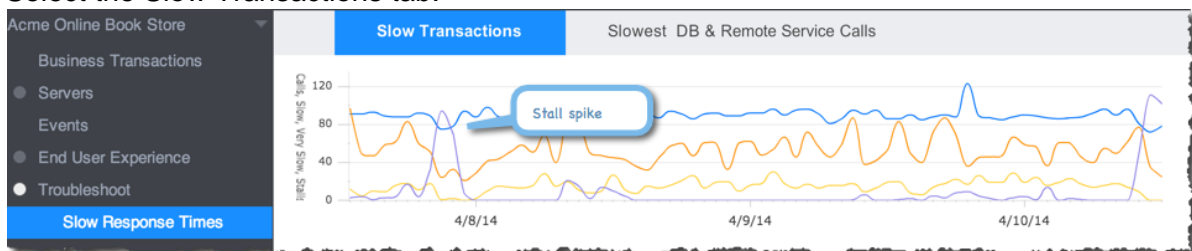
Set the Time Range

Use the Time Range dropdown menu to set the display to cover the timeframe needed.

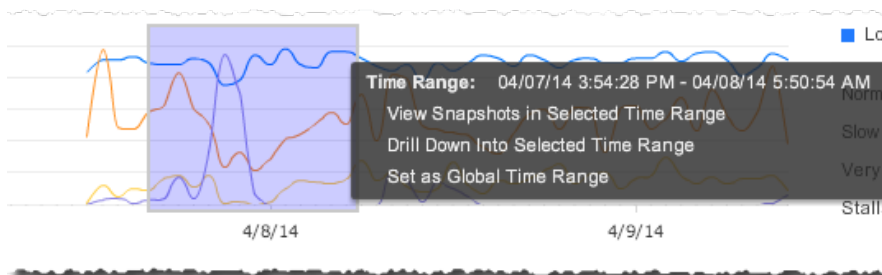


Drill Down to the Problem

1. Click Troubleshoot->Slow Response Times
2. Select the Slow Transactions tab.



3. Drag your mouse to select the the range where the problem occurred - in the example, the spike in stalls.
A Time Range popup appears.



4. Select Drill Down Into Selected Time Range.

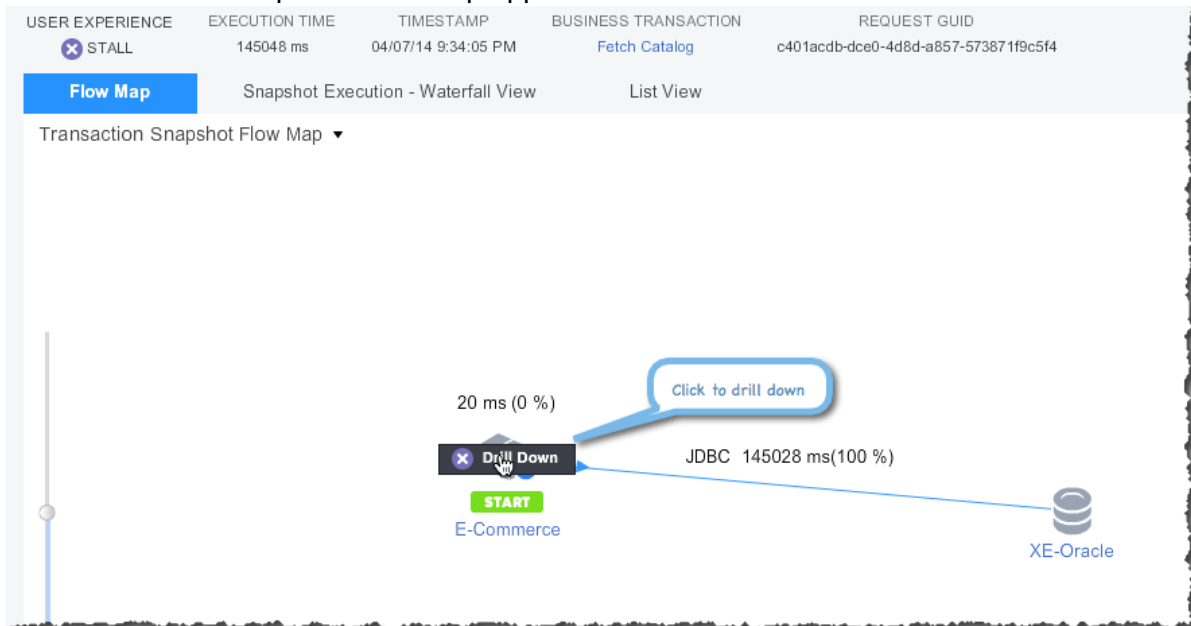
The Time Range Drill Down Workbench appears.

	Time	Exe Time (ms)	URL	Business Transaction
✕	04/07/14 11:04:02 PM	46257	/appdynamicspilot/ViewItems.action	Fetch Catalog
✕	04/07/14 11:04:01 PM	45370	/appdynamicspilot/ViewItems.action	Fetch Catalog
✕	04/07/14 11:02:11 PM	46906	/appdynamicspilot/ViewItems.action	Fetch Catalog
✕	04/07/14 11:02:07 PM	50171	/appdynamicspilot/ViewItems.action	Fetch Catalog
!	04/07/14 10:57:28 PM	10753	/appdynamicspilot/ViewCart/sendItems.action	Checkout
✕	04/07/14 10:45:10 PM	52037	/appdynamicspilot/ViewItems.action	Fetch Catalog

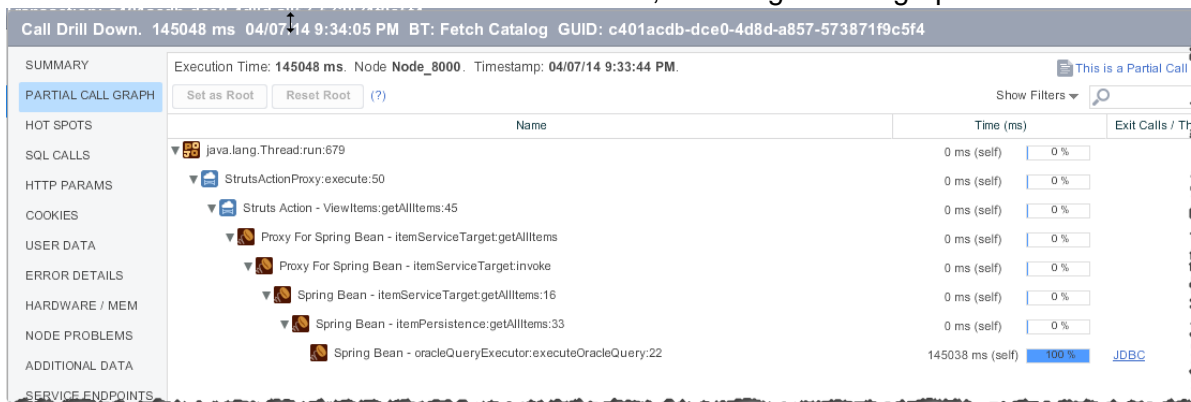
5. Select the Transaction Snapshots tab and the Slow and Error Transactions tab. Those stalled transactions all seem to be for the Fetch Catalog business transaction.
6. Click the Exe Time column to find the very slowest of the stalled transactions and click View Transaction Snapshot.

	Time	Exe Time (ms)	URL	Business Transaction
✕	04/07/14 9:34:05 PM	145048	/appdynamicspilot/ViewItems.action	Fetch Catalog
✕	04/07/14 9:33:59 PM	143873	/appdynamicspilot/ViewItems.action	Fetch Catalog
✕	04/07/14 9:34:03 PM	143607	/appdynamicspilot/ViewItems.action	Fetch Catalog
✕	04/07/14 9:34:20 PM	140380	/appdynamicspilot/ViewItems.action	Fetch Catalog
✕	04/07/14 9:33:25 PM	140192	/appdynamicspilot/ViewItems.action	Fetch Catalog

The Transaction Snapshot Flow Map appears.



7. Click Drill Down to see the Call Drill Down window, including the call graph.

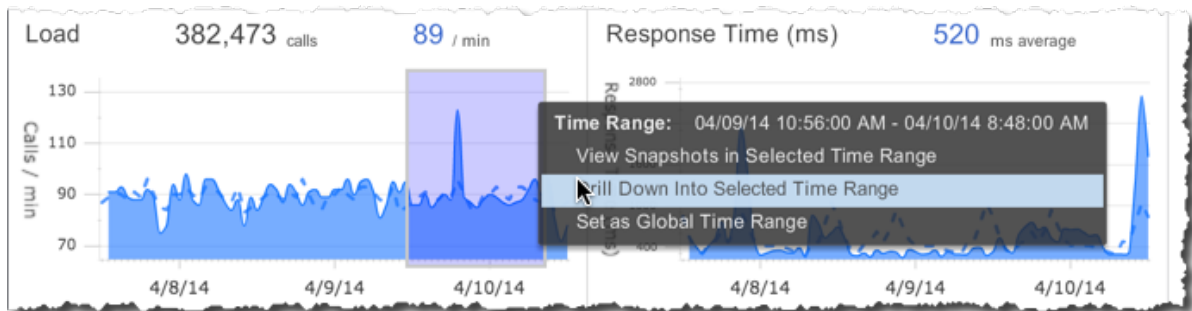


A slow database call was at the root of the problem.

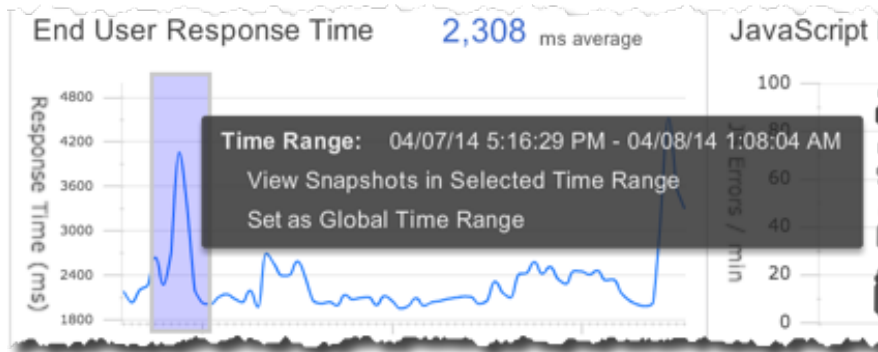
Changing Other Time Windows

You can use the mouse select time range method in many places throughout the UI. For example, in KPI listings in various dashboards:

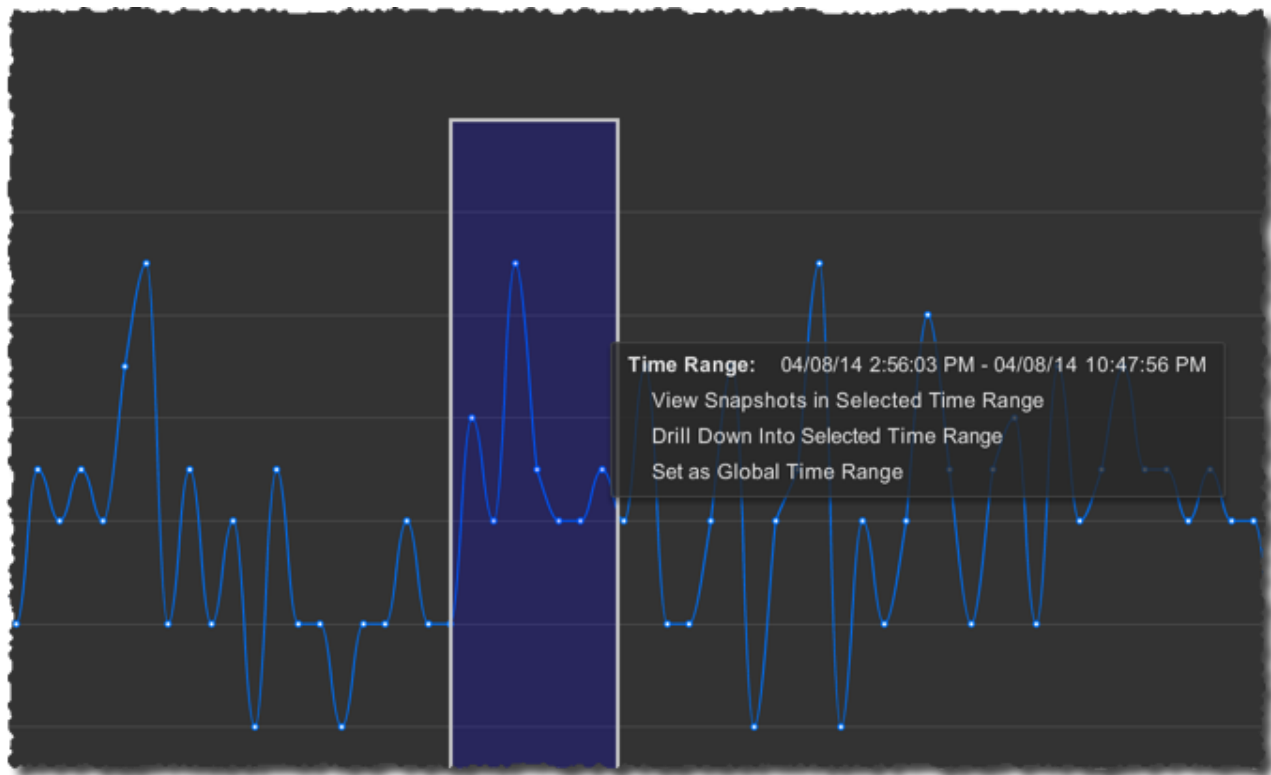
- Drill into Load from the top level dashboard



- Select snapshots occurring during a particular range in End User Response Time from the Web EUM dashboard



You can even use it in the Metric Browser:



Baselines

Whether or not some event in the past has been defined as an issue often depends on the *baseline* being used in the evaluation of that event. To understand how baselines work and what impact they can have, see [Behavior Learning and Anomaly Detection](#).

Learn More

- [KPI Graphs](#)
- [Rapid Troubleshooting](#)

Diagnostic Sessions

- [Triggering a Diagnostic Session](#)
 - [On-demand Diagnostic Sessions](#)
 - [To Start a Diagnostic Session for a Business Transaction Manually](#)
 - [Automatic Diagnostic Sessions For Slow and Error Transactions](#)
 - [To Configure Diagnostic Session Thresholds](#)
- [Accessing a Diagnostic Session](#)
 - [To View a Diagnostic Session](#)
- [Learn More](#)

This topic explains what a diagnostic session is and how to capture diagnostic session on a transaction.

A diagnostic session captures detailed data about the processing of a transaction as transaction snapshots over a defined period of time. This data includes full call graphs.

Diagnostic sessions can be triggered manually through the user interface or configured to start automatically when thresholds for slow, stalled, or error transactions are reached. If the diagnostic session is triggered manually, the diagnostic session collects snapshots on all the nodes that the selected business transaction passes through. If the diagnostic session is triggered to start automatically, the diagnostics session collects snapshots on the triggering node.

Triggering a Diagnostic Session

You can manually trigger a diagnostic session when you need one or configure them to start up automatically.

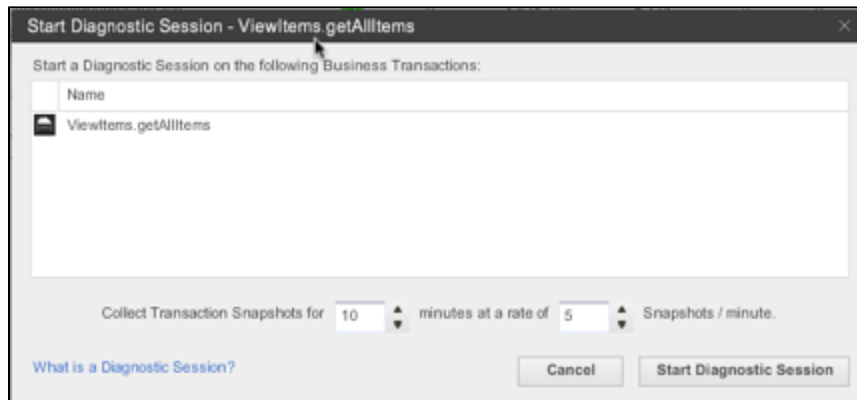
On-demand Diagnostic Sessions

On-demand diagnostic sessions must be started manually.

To Start a Diagnostic Session for a Business Transaction Manually

1. Display the dashboard for the business transaction that you want to analyze. See [Business Transactions List](#).

2. Click **Actions -> Start Diagnostic Session** or right-click on the selected business transaction and then click **Start Diagnostic Session**.
3. Specify the snapshot collection duration at the bottom of the Start Diagnostic Session window.



AppDynamics will start collecting transaction snapshots for that business transaction.

Automatic Diagnostic Sessions For Slow and Error Transactions

AppDynamics provides default thresholds to detect slow, very slow, stalled and error transactions. You can configure settings for triggering diagnostic sessions for these transactions.

To Configure Diagnostic Session Thresholds

1. In the left navigation pane, click **Configure -> Slow Transaction Thresholds**.
2. For configuring thresholds for business transactions click the **User Transaction Thresholds** tab.
 - a. For thresholds for background tasks, click the **Background Tasks Thresholds** tab.
3. In the thresholds tree list, select the scope of the threshold, either:
 - Default Thresholds
 - or
 - Individual Transaction Thresholds
4. In the right panel configure thresholds for when diagnostic sessions will be started.

Here you can set a trigger based on the percentage of requests that exceed the Slow Request threshold. For performance reasons you may not want to trigger a diagnostic session each time a threshold is exceeded.

5. Configure diagnostic session duration and collection frequency. This includes:
 - Number of snapshots to collect over a specified time period
 - Number of unsuccessful attempts per minute
 - Wait period between sessions

For performance reasons you want to limit the duration and frequency of diagnostic sessions to the minimum required time to obtain the maximum amount of information for troubleshooting purposes.

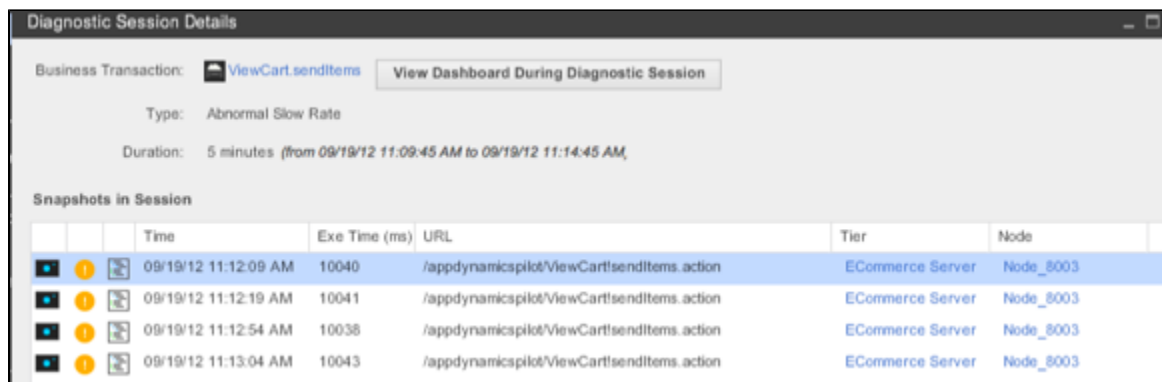
When there are ongoing performance problems you do not want a diagnostic session to run continuously. You can set a wait period between sessions and increase the time as needed.

Accessing a Diagnostic Session

You can access a diagnostic session from the transaction snapshot list.

To View a Diagnostic Session

1. Navigate to the transaction snapshot list.
See [To View Transaction Snapshots](#).
2. Select a transaction snapshot from the list and either double-click it or click **View Dashboard**.
3. Click the **Diagnostic Sessions** tab.
4. From the list select a diagnostic sessions and click **View Diagnostic Session**.
From there you can double-click a transaction snapshot to dive deeper.



Diagnostic Session Details						
Business Transaction: ViewCart.sendItems View Dashboard During Diagnostic Session						
Type: Abnormal Slow Rate						
Duration: 5 minutes (from 09/19/12 11:09:45 AM to 09/19/12 11:14:45 AM)						
Snapshots in Session						
		Time	Exe Time (ms)	URL	Tier	Node
		09/19/12 11:12:09 AM	10040	/appdynamicspilot/ViewCart/sendItems.action	ECommerce Server	Node_8003
		09/19/12 11:12:19 AM	10041	/appdynamicspilot/ViewCart/sendItems.action	ECommerce Server	Node_8003
		09/19/12 11:12:54 AM	10038	/appdynamicspilot/ViewCart/sendItems.action	ECommerce Server	Node_8003
		09/19/12 11:13:04 AM	10043	/appdynamicspilot/ViewCart/sendItems.action	ECommerce Server	Node_8003

Learn More

- [Transaction Snapshots](#)
- [Actions](#)
- [Diagnostic Actions](#)

Call Graphs

- [Call Graphs](#)
 - [To view call graphs](#)
 - [To Filter and Search for a Class Name in a Call Graph](#)
- [Diagnosing the Root Cause of Problems Using Call Graphs](#)
 - [Troubleshoot Problems using Call Graphs](#)
 - [Understanding Data Captured in Call Graphs](#)
 - [Class/Method Names and Time spent](#)
 - [External Calls Invoked by a Method](#)
 - [Viewing Call Graph Hot Spots](#)
 - [Advanced Options](#)
 - [Classes displayed in the call graphs](#)
 - [Packages and Namespaces excluded from the call graph](#)
 - [Configure instrumentation](#)
- [Learn More](#)

This topic describes call graphs for code-level diagnostics.

Call Graphs

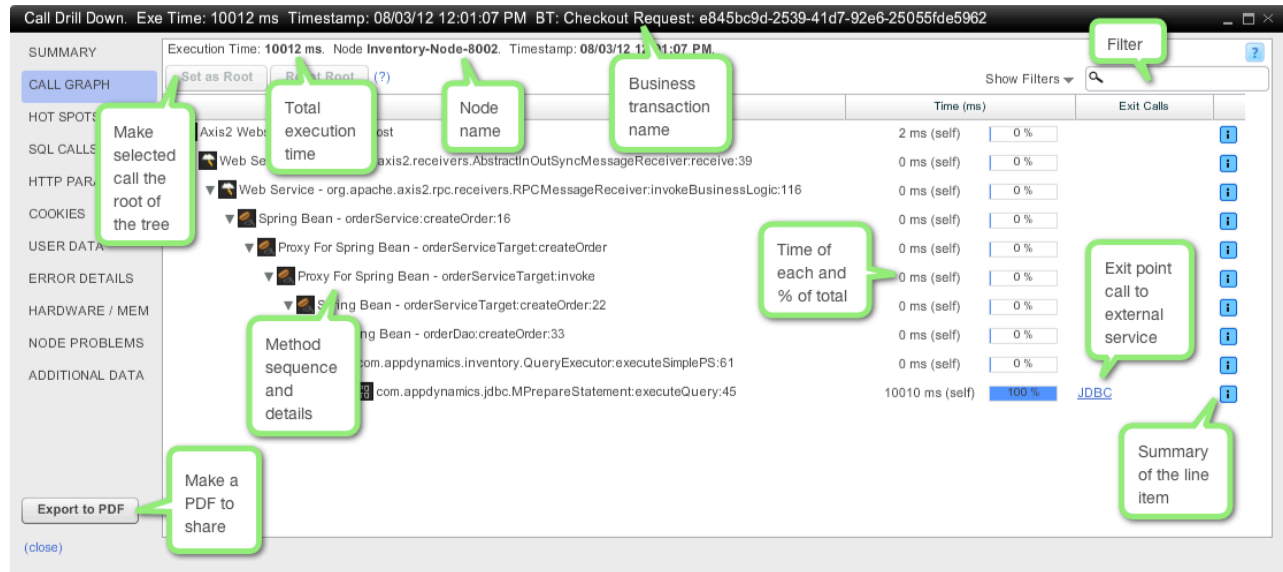
A call graph lists the methods in a call stack and provides information about each call. Call graphs help you diagnose performance issues and optimize the flow of a complex business transaction.

AppDynamics call graphs provide the following information:

- **Call information:** This includes the total execution time, the node name for the call graph, and the time stamp for the start of execution.
- **Method execution sequence:** An easily readable method execution sequence with the class names and the method names of each method.
 - **Application component information:** Application component information such as Servlet names, Struts Action classes, EJB names, Spring Bean IDs and proxies, message listeners, etc. POJOs are represented by class name, unless the POJO is a special type of component such as a struts action invoker. The class name is available for all call graph elements including those having logical class names.
 - **Code line number:** The line number of the element in the source code, if available.
- **Time distribution for each method:** The time distribution for each method along with the percentage of the total time in the call graph.
- **Exit call information:** Links to more information about exit calls such as JDBC, JMS, and Web Services.
- **Application component summary information:** Summary information for each element in the call graph.

In addition you can:

- **Filter the call graph list** using criteria such as application components, time, or all or part of the name.
- **Select an element of the list and set it as the root of the list** to drill down into large call graphs more efficiently.
- **Export the call graph to PDF** to share the information with other team members.



⚠ There are two kinds of call graphs: full and partial. Full call graphs capture the entire call sequence and are captured periodically for general monitoring purposes. Partial call graphs capture the call sequence from the point at which the call sequence has been determined to be slow or have errors, and thus may not include the initial steps in the sequence.

To view call graphs

1. From a dashboard, click the **Transaction Snapshots** tab.
2. Do one of the following:
 - Click **All Snapshots** to see all snapshots.
 - Click **Slow and Error Transactions** to see snapshots for slow and error transactions.
 - Click **Periodic Collection** to see snapshots collected periodically.
2. Select a particular transaction snapshot and click **View Transaction Snapshot**.
3. Click the **Drill Down** icon to see the call graph for the snapshot. By default the originating call graph in a business transaction, generated by the entry point on the entry point tier, displays.

To Filter and Search for a Class Name in a Call Graph

The call graph is displayed when you click on the **drill down** option on the snapshot. You can filter and search for a particular class name on the call graph. To do this, use the search box available on the left side of the call graph.

Diagnosing the Root Cause of Problems Using Call Graphs

Troubleshoot Problems using Call Graphs

You can use call graphs to troubleshoot the problems in the flow map. Drill down into each tier using the **Drill Down** option. If there is only one invocation, AppDynamics displays the call graph.

When there are multiple invocations for that tier which participated in

the transaction, the **Action -> Drill Down** menu displays a list of all call graphs for the node. Each call graph represents a call into the node. For example, if JVM A makes two remote calls to JVM B, then JVM B will have two call graphs.

Understanding Data Captured in Call Graphs

Class/Method Names and Time spent

Call graphs represent the sequence of execution of code on the application server. Each row represents a method call and the tree represents the execution sequence.

The total time spent in the call graph is displayed on the top left corner of the call graph.



The Time (ms) column in the call graph shows the time spent in each method.

External Calls Invoked by a Method

If a method invokes external calls outside of the app server, such as a JDBC query or a Web Service call, there is a link to the call in the call graph.

The following example shows the details of a JDBC call that include the query, the execution time of the query, the number of times that the application code iterated over the query results (ResultSet Count) and the time in milliseconds that the application code spent iterating over the results (ResultSet Time).



If the detail screen has a Drill Down link, you can get a call graph for the calls downstream from the original call.

Viewing Call Graph Hot Spots

The most expensive methods are listed in the Hot Spots section of

the transaction snapshot.

Advanced Options

Classes displayed in the call graphs

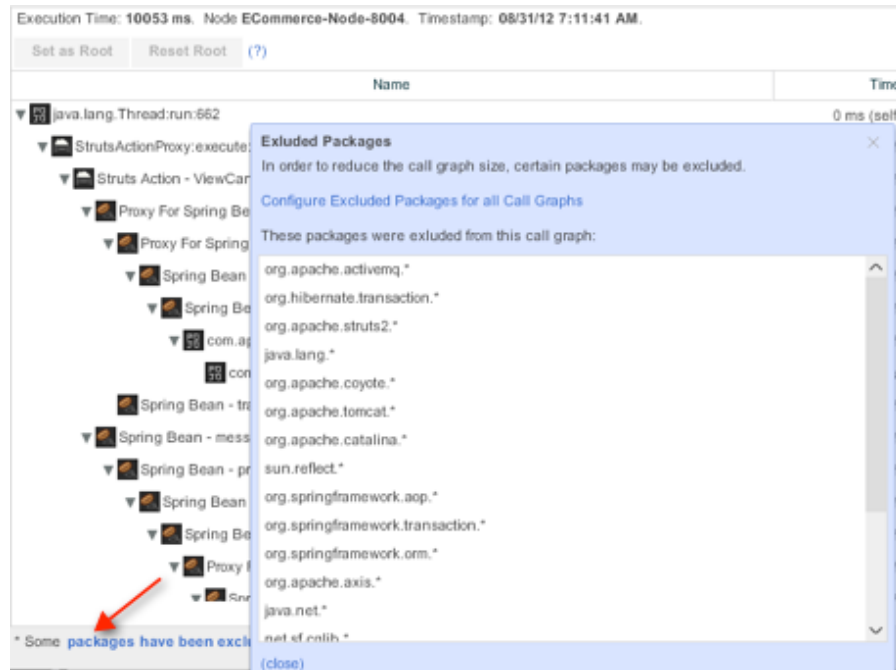
The sequence of Java method invocations in a call graph can include hundreds of classes. These classes are categorized as:

- Application classes
- AppServer/Container/Middleware classes (For example: Tomcat runtime classes, Websphere runtime classes, etc.)
- JDBC Driver/External infrastructure library classes (For example: Oracle Driver classes, Axis Web Service runtime classes, Hibernate classes, etc.)
- Third party utility libraries (For example: Apache commons libraries)
- Java/J2EE core libraries

Adding all of these classes together in a call graph is counterproductive for troubleshooting. The only mandatory classes required, are the classes from first category- the Application classes. You might require the other classes only in certain situations. To make the call graph more readable and to avoid the overhead of processing the timing information for all the non-application classes, other classes are not shown in the call graph.

Packages and Namespaces excluded from the call graph

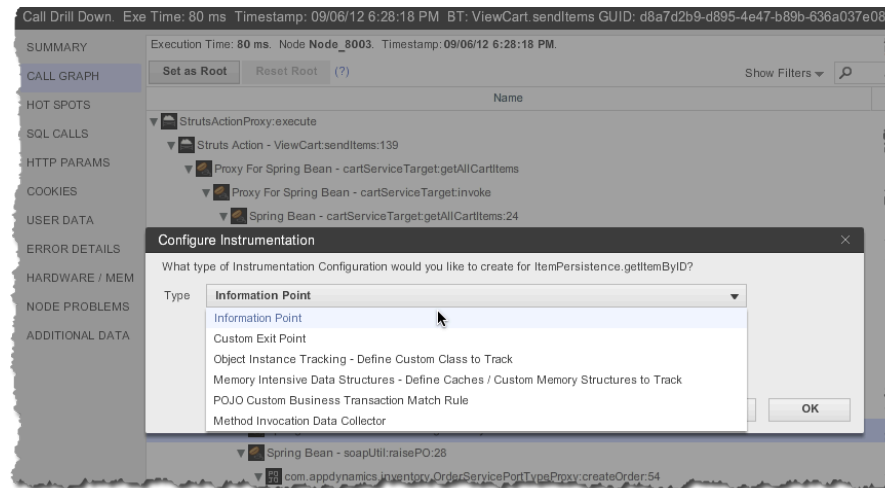
A call graph can have a list of packages (Java) or namespaces (.NET) that are not displayed. To access this list, click on the message for "Some packages/namespaces have been excluded from the Call graph" message.



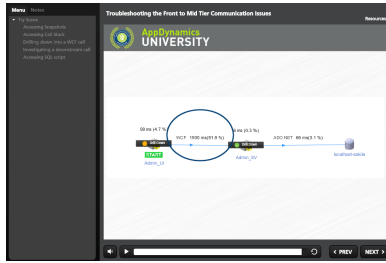
The PHP and Node.js agents do not support call graph exclusion.

Configure instrumentation

You can right-click on any item in a call graph and select **Configure Instrumentation for this Class/Method**.



The Configure Instrumentation window presents a drop-down menu form which you can select the type of configuration that you want to create for the method.



This short interactive video traces the typical steps of identifying the cause of communication issues between tiers in your application.

Learn More

- [Configure Call Graphs](#)
- [Configure Code Metric Information Points](#)
- [Configure Custom Exit Points for Java](#)
- [Configure and Use Object Instance Tracking for Java](#)
- [Configure Memory Monitoring for Java](#)
- [POJO Entry Points](#)
- [Configure Data Collectors](#)
- [Tracing Multi-Threaded Transactions \(Java\)](#)

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](#)
- [Aggressive Slow 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 **sub** tab for the framework you are configuring.

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 for Java or namespaces for .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**.

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.

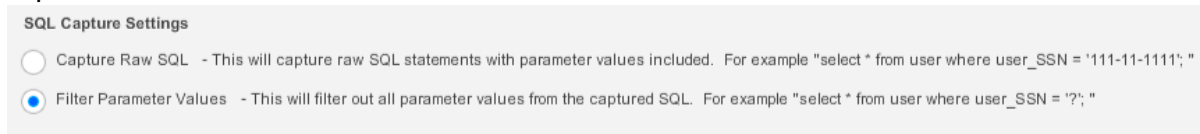
Normalizing a query prevents display of sensitive data, such as social security numbers or credit card numbers, which are potential query parameters. Normalizing queries also helps to organize SQL data by flattening the parameter values for the query so that the statistics from different executions of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

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

Note that SQL passed in as a string is not redacted.

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.



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

Slow Transaction Snapshot Collection

AppDynamics captures the full execution path of a request after its execution time exceeds the business transaction threshold for slow requests. 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.

Aggressive Slow Snapshot Collection (Java only)

For Java only, you can configure more aggressive snapshot collection to capture the full execution path of requests before thresholds are crossed.

By default the agent disables aggressive snapshot collection to minimize overhead. Enable it when you start to experience performance problems in order to diagnose the 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](#)

Analyze

AppDynamics provides tools to help you analyze patterns and discover relationships between different metrics and performance data over time. See:

Scalability Analysis

- [Scalability Analysis Comparisons](#)
 - [To access Scalability Analysis](#)
 - [To use scalability data](#)
- [Learn More](#)

Scalability problems in a distributed environment can cause remote communication overhead. Examples of scalability problems include:

- Increased inter-tier time when tiers are newly separated
- Increased inter-tier time in conjunction with chattiness
- Over-sized payloads with network saturation

Scalability Analysis Comparisons

You can compare:

- **Response Time vs Application Load**
You can compare average response time versus calls per minute for a business application, a business transaction, a tier, or a node.
- **CPU Utilization vs Application Load**
You can compare CPU % usage versus calls per minute for a business application, a tier, or a node.
To compare CPU % usage, a standalone or embedded machine agent must be installed on the node machine.

The following graph shows Response Time vs Load at the application level for the past three days.

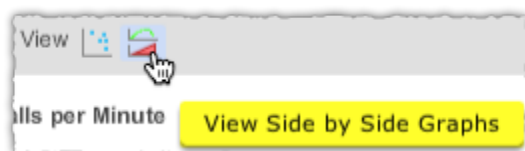


To access Scalability Analysis

In the left navigation pane for an application click **Analyze -> Scalability Analysis**.

To use scalability data

1. Select the time range to be covered by the analysis from the [Time Range](#) menu. Modifying the time range in this window does not modify the time range settings in other parts of the UI.
2. Click either the **Response Time vs Load** tab or the **CPU Utilization vs Load** tab, depending on which analysis you want to perform.
3. In the left panel of the scalability viewer either:
 - Select Application for analysis at the application level.
or
 - Navigate to the business transaction, tier, or node that you want to analyze.
4. Browse the graph and explore:
 - You can change between a scatter plot view and a side-by-side graph view.



- You can toggle the Best Fit Line in the scatter plot view. The Best Fit Line is calculated using the Quadratic Least Squares algorithm.
- You can hover over data points to view metrics at particular times.
- Double-click on a data point to open its metrics in the [Metric Browser](#).

- Click the tab header **Analyze Response Time Vs Load** or **Analyze CPU vs Load** to update the graph.
- You can export the data as a comma-separated values file:



Learn More

- [Infrastructure Metrics](#)
- [Metric Browser](#)

Correlation Analysis

- [To perform correlation analysis between two metrics](#)

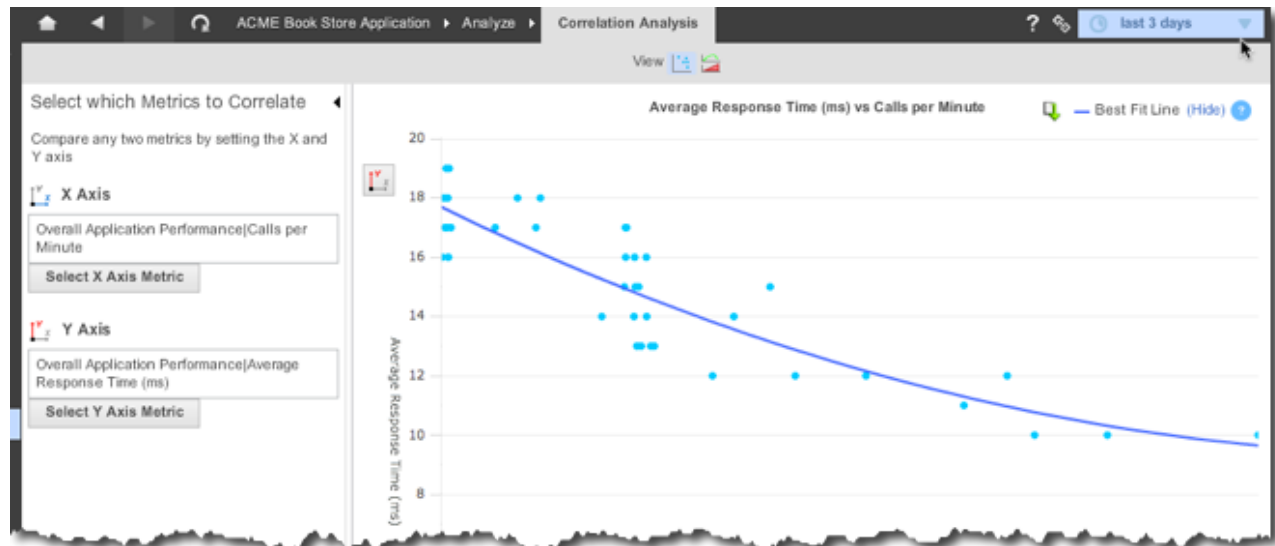
Learn More

Correlation analysis lets you compare two metrics on different axes to see how one metric correlates with the other.

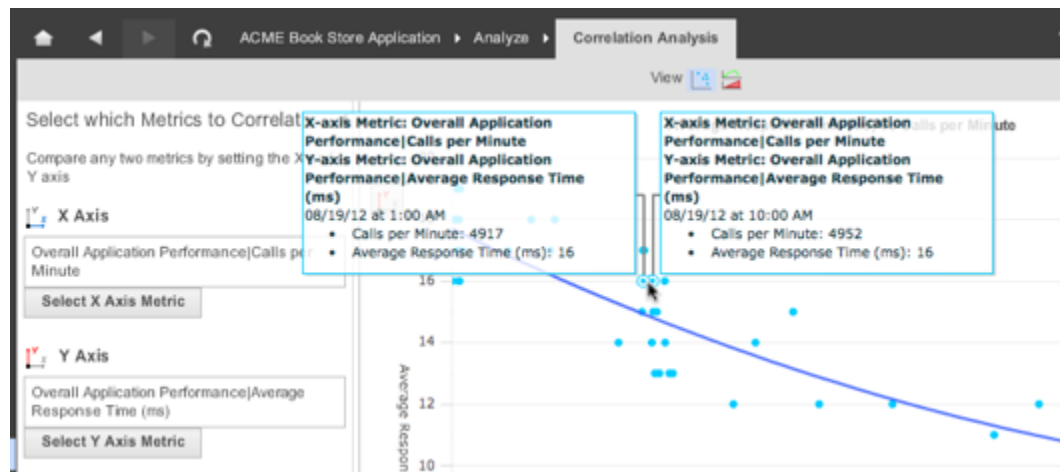
To perform correlation analysis between two metrics

1. In the left navigation pane, click **Analyze -> Correlation Analysis**.
2. Click **Select X Axis Metric**.
3. From the metric browser navigate to the metric that you want to graph on the X axis and double-click the metric.
4. Click **Select Y Axis Metric**.
3. From the metric browser navigate to the metric that you want to graph on the Y axis and double-click the metric.

The graph is displayed when metrics for both the axes are selected.



Hover over one or more of the data points to view their metrics at particular times.



Learn More

- [Metric Browser](#)

Compare Releases

- [Entities to Compare](#)
- [Metrics to Compare](#)
 - [To Compare Releases](#)
- [Learn More](#)

You can compare metrics for two different time periods on a split screen.

This feature is particularly useful for comparing different versions of a release.

Entities to Compare

You can compare summary metrics:

- for the entire application

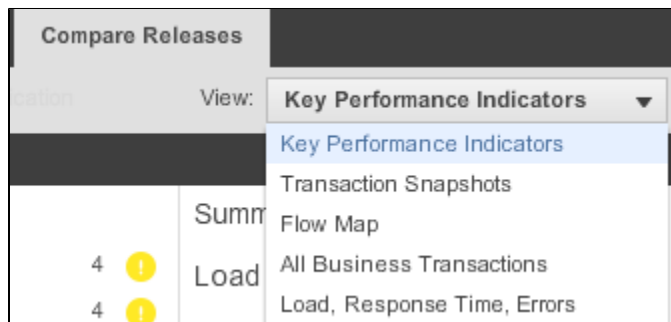
- for a specific business transaction
- for a specific tier
- for a specific node

Metrics to Compare

The metrics you can compare are:

- **Summary Key Performance Indicators (KPIs)**
Displays the KPI summaries from the dashboards for the selected entity (application/business transaction/tier/node).
- **Transaction Snapshots**
Displays the transaction snapshots for the selected entity.
- **Graphical Flow**
Displays flow graphs for the selected entity.
- **KPI Trend Graphs**
Displays the KPI summaries from the dashboards for the selected entity (application/business transaction/tier/node).

Select which sets of metrics you want to compare from the View drop-down menu.



To Compare Releases

1. In the left navigation pane, click **Analyze -> Compare Releases**.
2. Set the time ranges to compare from the Time Range drop-down menus in both panes.
3. Select the entities to compare in the Select What to Compare panel.
4. Select the metrics that you want to compare from the View drop-down menu.
5. Click **Compare**.

Whenever you change the entities or the metrics, click **Compare** to refresh the display.

Learn More

- [Dashboards](#)
- [Transaction Snapshots](#)
- [Time Ranges](#)

Troubleshoot Health Rule Violations

- [Health and Health Rules](#)
- [Troubleshoot Health Rule Violations](#)
 - [To find all health rule violations](#)
 - [To troubleshoot a health rule violation](#)
 - [To see health rule status in the UI](#)
- [Learn More](#)

Health and Health Rules

"Health" throughout the AppDynamics UI refers to the extent to which the component being monitored is operating within the acceptable limits defined by health rules. Health rules allow you to automate pro-active monitoring and problem mediation in your managed environment. By default, AppDynamics provides a set of basic health rules, which you can extend, add to, or remove as your needs dictate.

A health rule violation exists when the conditions that define the rule are true. For example, you might have defined a health rule condition that states that a CPU%Busy rate of more than 90% on any node is a critical condition. If the rate on a node then goes over 90%, the health rule is said to "violate" and the AppDynamics UI displays a notification of that violation.

Because there is a set of default health rules, you may see health rule violations reported for your application even if you have not set up your own health rules. If you see violations reported for the APPDYNAMICS_DEFAULT_TXT business transaction, these are for default health rule violations in the All Other Traffic business transaction. If you are not interested in monitoring these business transactions, you may want to examine your business transaction setup. See [Organizing Traffic as Business Transactions](#).

For general information about health rules, see [Health Rules](#). For information on setting up your own health rules, see [Configure Health Rules](#).

Troubleshoot Health Rule Violations

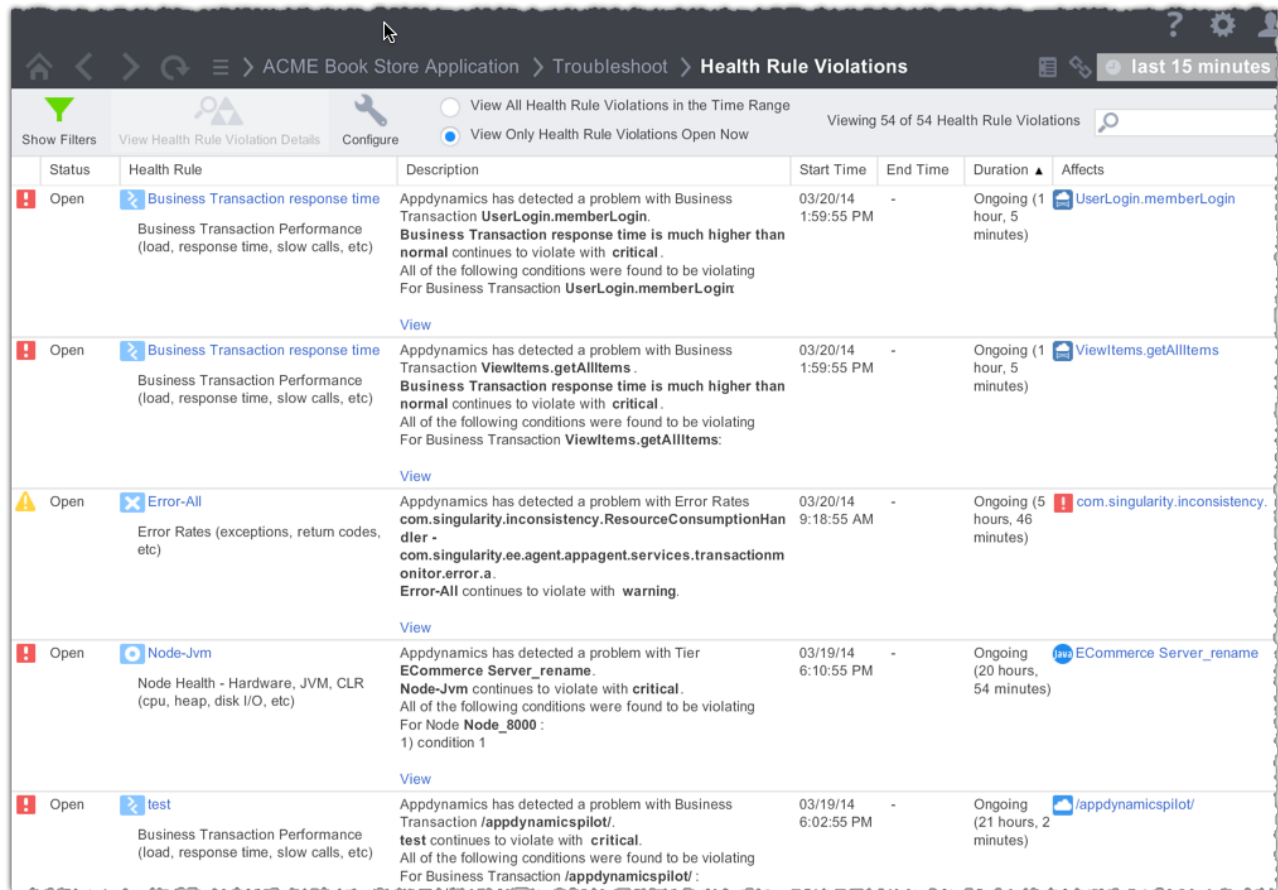
To start troubleshooting health rule violations, you can:

- [Get a list of all the health rule violations](#) by clicking **Troubleshoot -> Health Rule Violations**.
- Click on a particular health rule violation you see [displayed in the UI](#).

You can access the list of health rule violations in your application for the selected time range.

To find all health rule violations

1. In the left navigation pane, click **Troubleshoot -> Health Rule Violations**.
The list of health violations displays.



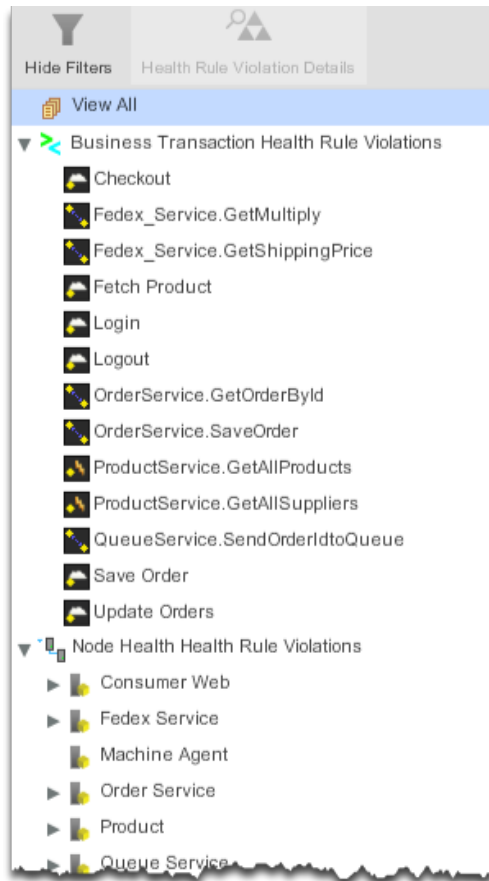
Status	Health Rule	Description	Start Time	End Time	Duration	Affects
Open	Business Transaction response time Business Transaction Performance (load, response time, slow calls, etc)	Appdynamics has detected a problem with Business Transaction UserLogin.memberLogin . Business Transaction response time is much higher than normal continues to violate with critical . All of the following conditions were found to be violating For Business Transaction UserLogin.memberLogin	03/20/14 1:59:55 PM	-	Ongoing (1 hour, 5 minutes)	UserLogin.memberLogin
Open	Business Transaction response time Business Transaction Performance (load, response time, slow calls, etc)	Appdynamics has detected a problem with Business Transaction ViewItems.getAllItems . Business Transaction response time is much higher than normal continues to violate with critical . All of the following conditions were found to be violating For Business Transaction ViewItems.getAllItems :	03/20/14 1:59:55 PM	-	Ongoing (1 hour, 5 minutes)	ViewItems.getAllItems
Open	Error-All Error Rates (exceptions, return codes, etc)	Appdynamics has detected a problem with Error Rates com.singularity.inconsistency.ResourceConsumptionHandler - com.singularity.ee.agent.appagent.services.transactionmonitor.error.a . Error-All continues to violate with warning .	03/20/14 9:18:55 AM	-	Ongoing (5 hours, 46 minutes)	com.singularity.inconsistency.
Open	Node-Jvm Node Health - Hardware, JVM, CLR (cpu, heap, disk I/O, etc)	Appdynamics has detected a problem with Tier ECommerce Server_rename . Node-Jvm continues to violate with critical . All of the following conditions were found to be violating For Node Node_8000 : 1) condition 1	03/19/14 6:10:55 PM	-	Ongoing (20 hours, 54 minutes)	ECommerce Server_rename
Open	test Business Transaction Performance (load, response time, slow calls, etc)	Appdynamics has detected a problem with Business Transaction /appdynamicspilot/ . test continues to violate with critical . All of the following conditions were found to be violating For Business Transaction /appdynamicspilot/ :	03/19/14 6:02:55 PM	-	Ongoing (21 hours, 2 minutes)	/appdynamicspilot/

2. Select **View All Health Rule Violations in the Time Range** or **View Only Health Rule Violations Open Now**.

It is possible that health rule violations that were reported are no longer open because remedial action has been taken or performance has improved on its own.

3. To see the filters click **Show Filters**. To hide them click **Hide Filters**.

With the filters showing in the left filters panel you can select the health rule violations that you want to troubleshoot. You can view all health rule violations or expand the nodes in the tree to select by health rule type (such as business transaction health rules or node health rules) or affected entity (such as business transaction, tier or node).



You can filter health rule violations by entering the name of the health rule in the search field on the upper right.

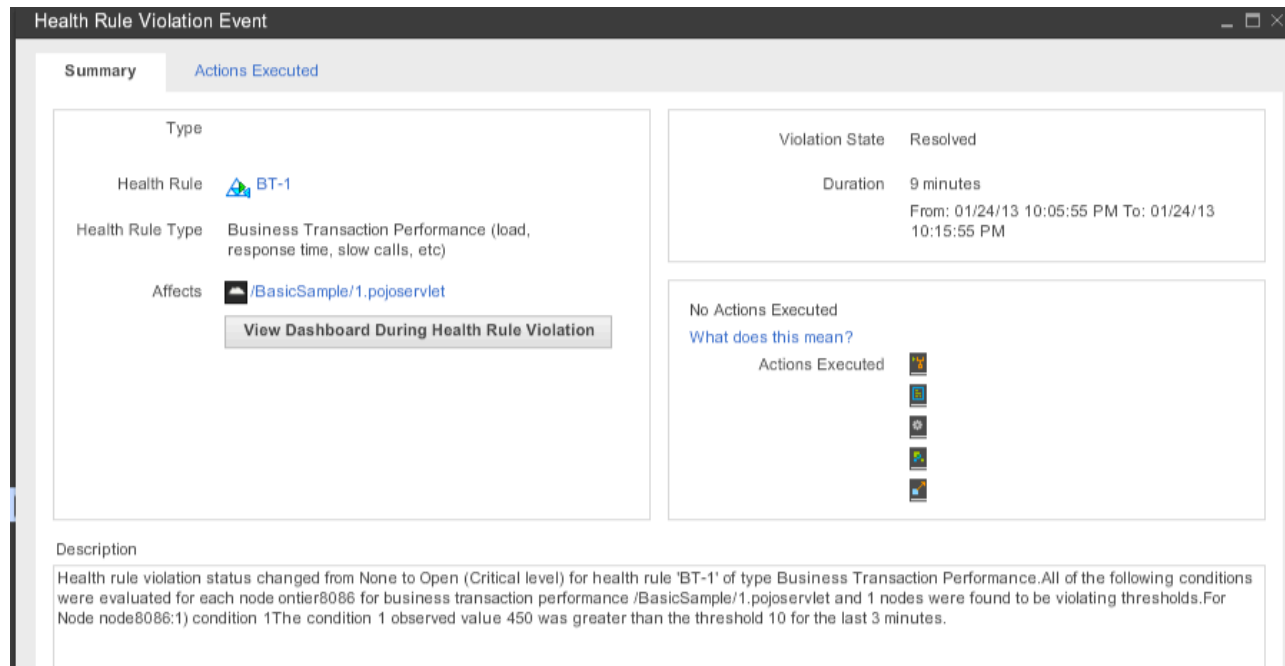
The health rule violations are displayed in the right panel, with their status, description, start time, end time and duration (if ended), and the affected entity.

To troubleshoot a health rule violation

Once you have located the violation you are interested in, you can get more information in three ways:

- To see the health rule definition that was violated for a specific violation, find the health rule violation in the list and in the Health Rule column, click the link to the health rule configuration. The Edit Health Rule window for the specific definition appears.
- To see the dashboard for the entity, such as a business transaction or a node, affected by the violation, click the link to the entity in the Affects column. The Transaction Flow Map appears.
- To troubleshoot a specific health rule violation, select the health rule violation row from the list and click **Health Rule Violation Details** in the top bar.


The Health Rule Violation Event window displays a summary of the violation and any actions that were executed to respond to it.




Health Rule Violation Event

Summary | [Actions Executed](#)

Type

Health Rule  BT-1

Health Rule Type Business Transaction Performance (load, response time, slow calls, etc)

Affects  /BasicSample/1.pojoservlet

[View Dashboard During Health Rule Violation](#)

Violation State Resolved

Duration 9 minutes

From: 01/24/13 10:05:55 PM To: 01/24/13 10:15:55 PM

No Actions Executed

[What does this mean?](#)

Actions Executed

Description

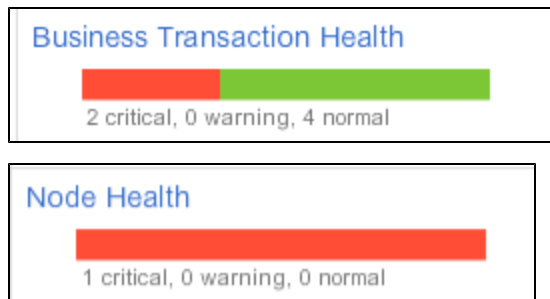
Health rule violation status changed from None to Open (Critical level) for health rule 'BT-1' of type Business Transaction Performance. All of the following conditions were evaluated for each node on tier8086 for business transaction performance /BasicSample/1.pojoservlet and 1 nodes were found to be violating thresholds. For Node node8086:1) condition 1The condition 1 observed value 450 was greater than the threshold 10 for the last 3 minutes.

You can click the **View Dashboard During Health Rule Violation** in the details window to view the dashboard at the time the violation occurred. The time range in this and all other dashboards is set to the time range of the health rule violation. From the dashboard you can get an overall picture of the application at the time of the violation. If you select the Transaction Snapshots tab you get a list of relevant snapshots which allows you to drill down to the root cause of the problem. See [Transaction Snapshots](#) for more information.

To see health rule status in the UI

Across the UI, health rule status is color-coded: green is healthy; yellow/orange is a warning condition; and red is a critical condition. If you see a health rule violation reported in the UI, you can click it to get more information about the violation.

Here are the health summary bars on the dashboards:



There is a health column in the business transaction list:

View Dashboard More Actions View Options			
	Name	Health	Server Time (ms)
	ViewItems.getAllItems	✓	33
	ViewCart.sendItems	⚠	756
	UserLogout.memberLogout	✓	10
	UserLogin.memberLogin	✓	16

In the Events panel on the dashboards, health violations are displayed as events.

Events

Health Rule Violations Started	2	⚠
Business Transaction Health	3	✗
Node Health	1	⚠
Error Rates	1	⚠
Code Problems	74	✗
Application Changes	4	!

To see a summary of the violation, click a health rule violation from the Events list, then select the violation you are interested in from the list that appears. Click **View Event Details** in the top bar and the Health Rule Violation Started window appears. It displays detailed information and a link to the appropriate dashboard at the time of the violation. If any Policy actions were executed in response to the violation, they are also displayed.

Health Rule Violation Started - Critical

Summary
Actions Executed (0)

Event Type
✗ Health Rule Violation Started - Critical

Health Rule
⚡ Slow Requests

Health Rule Type
Business Transaction Performance (load, response time, slow calls, etc)

Affects
🖨 Supplier Search

View Dashboard During Health Rule Violation

Violation State
Open

Duration
Ongoing (6 minutes)
From: 05/06/13 4:48:55 PM To: -

No Actions Executed
What does this mean?

Description

Health rule violation status changed from None to Open (Critical level) for health rule 'Slow Requests' of type Business Transaction Performance.

All of the following conditions were evaluated for each node on E-Commerce for business transaction performance Supplier Search and 1 nodes were found to be violating thresholds.

For Node E-Commerce-Node-8004:

Learn More

- Health Rules

- [Configure Health Rules](#)
- [Organizing Traffic as Business Transactions](#)
- [Policies](#)
- [Alert and Respond](#)
- [Transaction Snapshots](#)

Troubleshoot Java Application Problems

Troubleshoot Slow Response Times for Java

- [How Do You Know Response Time is Slow?](#)
- [Troubleshooting Steps](#)
 - [Step 1 - Slow or stalled business transactions?](#)
 - [Step 2 - Slow DB or remote service calls?](#)
 - [Step 3 - Affects 1 or more nodes?](#)
 - [Step 4 - Backend problem?](#)
 - [Step 5 - CPU saturated?](#)
 - [Step 6 - Significant garbage collection activity?](#)
 - [Step 7 - Memory leak?](#)
 - [Step 8 - Resource leak?](#)
 - [None of the above?](#)

How Do You Know Response Time is Slow?

There are many ways you can learn that your application's response time is slow:

- You received an email or SMS alert from AppDynamics (see [Alert and Respond](#)). The alert provides details about the problem that triggered the alert. If the problem is related to slow response time, start troubleshooting at [Step 1](#).
- Someone reported a problem such as "it's taking a long time to check out" or "the app timed out when I tried to add an item to the cart."
The problem is slow response time related to one or more business transactions. Start troubleshooting at [Step 1](#).
- A custom dashboard shows a problem. If the problem is related to slow response time, start troubleshooting at [Step 1](#).
- You are looking at the Application Dashboard for a business application, shown below:



1. Look at the traffic flow lines in the flow map, the Business Transaction Health pane, the Transaction Scorecard pane, and the Response Time graph. If you see problems (yellow or red lines, spikes in response time), the problem is slow response time, and is probably related to your AppDynamics business application. Start troubleshooting at [Step 1](#).
2. Look at the Tier icons in the flow map. If you see yellow or red, you have a problem with one or more nodes, which may or may not result in slow response time. Start troubleshooting at [Step 3](#).
3. Look at the Events pane. If you see red or yellow, the problem reflects a change in application state that is of potential interest, which may or may not result in slow response time. See [Tutorial for Java - Troubleshooting using Events](#).
4. Look at the Server Health pane. If you see red or yellow, the problem reflects a server health rule violation, which may or may not result in slow response time. See [Tutorial for Java - Server Health](#).

Need more help?

- [Application Dashboard](#)
- [Flow Maps](#)

Troubleshooting Steps

The following steps help you determine whether your problem is related to your business application or to your hardware or software infrastructure. Each step shows you how to display detailed information to help you pinpoint the source of the problem and quickly resolve it.

**Step 1 -
Slow or
stalled
business
transaction
s?**

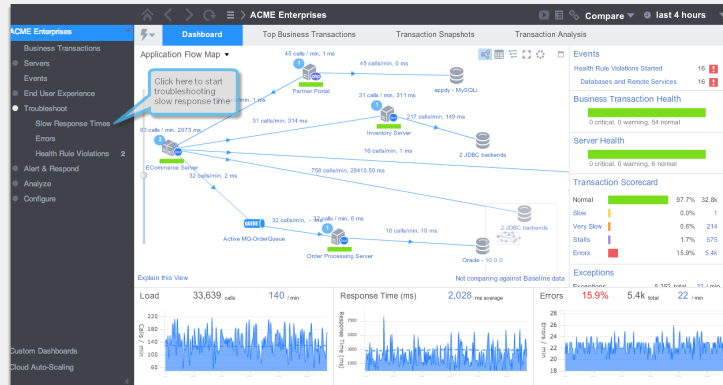
Are there any slow or stalled business transactions?

▼ How do I know?

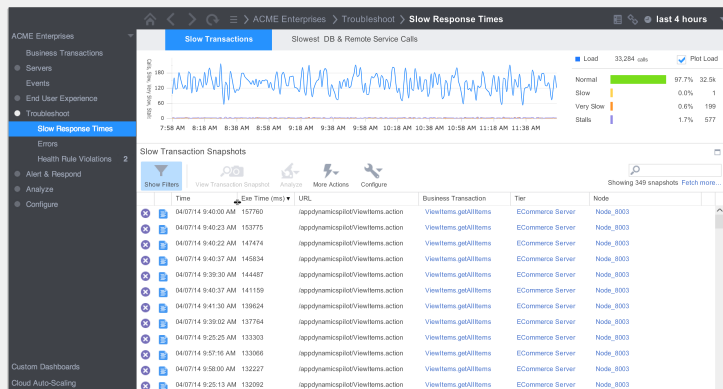
How do I know if business transactions are slow or stalled?

1. Click **Troubleshoot** -> **Slow Response Times**

You can also access this information from tabs in the various dashboards.



2. Click the **Slow Transactions** tab if it is not selected.



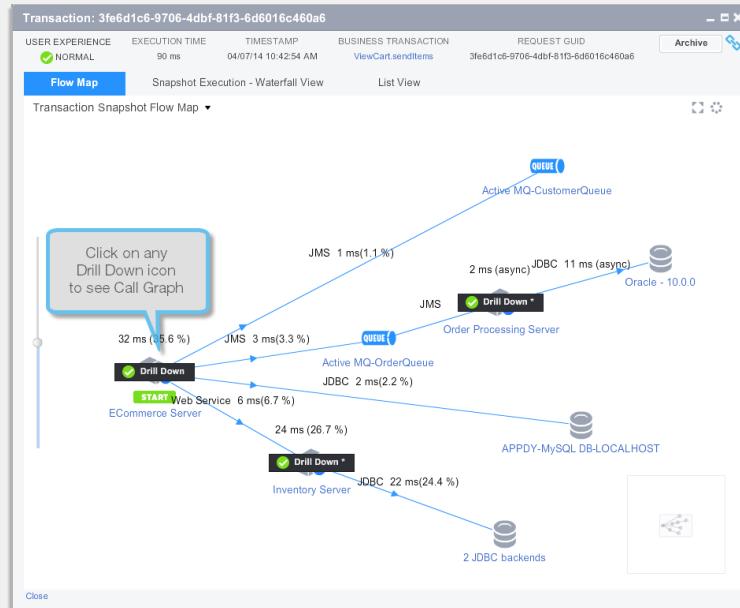
- In the upper pane AppDynamics displays a graph of the slow, very slow, and stalled transactions for the time period specified in the Time Range drop-down menu. If the load is not displayed, you can click the Plot Load checkbox at the upper right to see the load.
- In the lower pane AppDynamics displays the transaction snapshots for slow, very slow, and stalled transactions.

If you see one or more slow transaction snapshots on this page, the answer to this question is Yes. Otherwise, the answer is No.

No – Go to **Step 2**.

Yes – You have one or more slow or stalled transactions, and need to drill down to find the root cause.

1. In the lower pane of the Slow Transactions tab, click the Exe Time column to sort the transactions from slowest to fastest.
2. Select a snapshot from the list and click View Transaction Snapshot. You see the Transaction Flow Map. Choose the Flow Map tab if it is not already selected.



3. Click a Drill Down icon to display a call graph for a problematic part of the transaction. Once you are in the call graph you can look for methods that have a significant response time. In this example, the executeQuery method is responsible for 54.5% of response time.

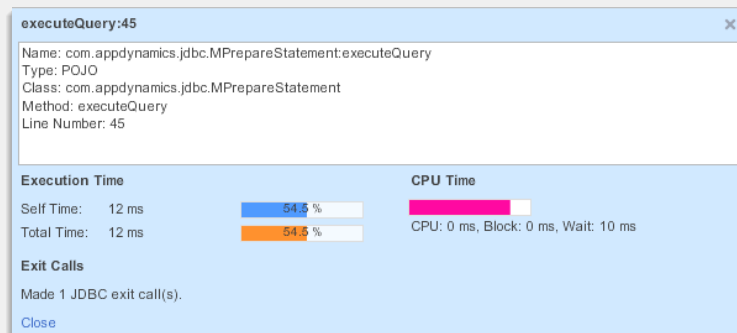
Call Drill Down: 22 ms 04/07/14 10:42:55 AM BT: ViewCart.sendItems GUID: 3fe6d1c6-9706-4dbf-81f3-6d6016c460a6

Execution Time: 22 ms. Node: Node_0002. Timestamp: 04/07/14 10:42:55 AM.

Name	Time (ms)	CPU time	Exit Calls
Web Service - org.apache.axis2.receivers.AbstractOutboundMessageReceiver.receive.39	2 ms (self)	5.1%	1
Web Service - org.apache.axis2.receivers.RPCMessageReceiver.invokeBusinessLogic.115	0 ms (self)	0%	1
Spring Bean - orderService.createOrder.16	0 ms (self)	0%	1
Proxy For Spring Bean - orderServiceTarget.createOrder	0 ms (self)	0%	1
Spring Bean - transactionManager.doBegin.510	1 ms (self)	4.5%	1
Spring Bean - dataSource.getConnection.880	1 ms (self)	4.5%	1
Proxy For Spring Bean - orderServiceTarget.invoke	0 ms (self)	0%	1
Spring Bean - orderServiceTarget.createOrder.22	3 ms (self)	13.8%	1
com.appdynamics.inventory.QueryExecutor.executeQuery.45	12 ms (self)	54.5%	1
com.appdynamics.jdbc.MPreparedStatement.executeQuery.578	2 ms (self)	9.1%	1
com.ctc.westaw.BaseStreamWriter.flush.269	0 ms (self)	0%	1
com.ctc.westaw.BufferingWriter.flush.184	0 ms (self)	0%	1
com.ctc.westaw.UTFWriter.flush.92	1 ms (self)	4.5%	1

This query is taking 54.5% of the execution time

- Click the Information icon in the right column to see more details. Provide this information to the personnel responsible for addressing this issue.



If there are multiple slow or stalled transactions, repeat this step until you have resolved them all. However, there may be additional problems you haven't resolved. Continue to [Step 2](#).

Need more help?

- [Transaction Snapshots](#)

Step 2 - Slow DB or remote service calls?

Is there one or more slow DB or remote service call?

How do I know?

How do I know if I have slow DB or remote service calls?

1. Click **Troubleshoot -> Slow Response Times**, then click the **Slowest DB & Remote Service Calls** tab if it is not selected.

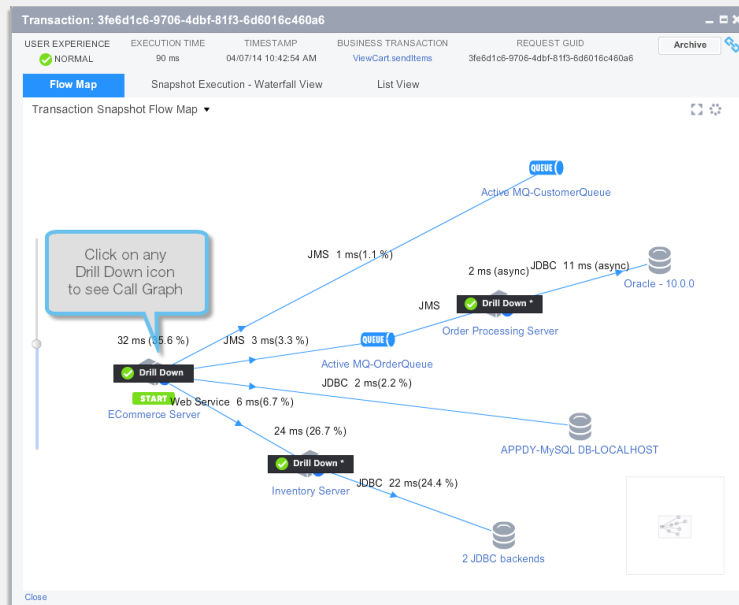
Slow Transactions		Slowest DB & Remote Service Calls			
Call Type		These are the calls with largest observed individual execution time (Max Time) during the specified time range.			
	Call	Avg. Time per Call	Number of Calls	Max Time (ms)	Snapshots
All Calls	SELECT COUNT(1) COUNT FROM ITEM IT1, ITEM IT2	113745.5	576	157728	View snapshots
JDBC	ORDERSERVICE.CREATEORDER	8660	242	10050	View snapshots
JMS	INSERT INTO ORDERREQUEST (ITEM_ID, NOTES) VALUES (?, ?)	302.7	6730	10001	View snapshots
DB	ORDERQUEUE	130	1	130	View snapshots
HTTP	DELETE FROM CART	0.6	7807	69	View snapshots
	INSERT INTO CART (ITEM_ID, USER_ID) VALUES (7, 7)	0.5	7423	80	View snapshots
	INSERT INTO ORDERS (QUANTITY, CREATEDON, ITEMID) VALUES	0.5	6180	61	View snapshots
	DB TRANSACTION COMMIT	0.4	3475	38	View snapshots
	PREPARED STATEMENT BATCH	0.4	6204	38	View snapshots
	SELECT ITEM0_ID AS ID0_0, ITEM0_QUANTITY AS QUANTITY0_0	0.4	5654	37	View snapshots
Call Details		Correlated Snapshots			
		SELECT COUNT(1) COUNT FROM ITEM IT1, ITEM IT2			

If you see one or more slow calls on this page, the answer to this question is Yes. Otherwise, the answer is No.

No – Go to [Step 3](#).

Yes – You have one or more slow DB or remote service calls, and need to drill down to find the root cause.

1. In the Call Type panel select the type of call for which you want to see information, or select All Calls.
2. Sort by Average Time per Call to display the slowest calls at the top of the list.
3. To see transaction snapshots for the business transaction that is correlated with a slow call, you can:
 - Click the **View Snapshots** link in the right column to display correlated snapshots in a new window.
 - Select the call and click the **Correlated Snapshots** tab in the lower panel to display correlated snapshots at the bottom of the screen.
4. Select a snapshot from the list and click View Transaction Snapshot. Choose the Flow Map tab if it is not already selected, then click a Drill Down icon to display a call graph for a problematic part of the transaction.



5. Once you are in the call graph you can look for methods that have a significant response time. In this example, an Oracle query is responsible for 99.7% of response time. Provide this information to the personnel responsible for addressing this issue.

Transaction: 9144f609-f716-4fa3-815f-23f473247a69		
Call Call Down, Exe Time: 9968 ms, Timestamp: 03/27/14 4:49:37 PM, BT: Fetch Catalog GUID: 9144f609-f716-4fa3-815f-23f473247a69		
SUMMARY	Execution Time: 9968 ms, Node: E-Commerce-Node-0000, Timestamp: 03/27/14 4:50:27 PM	
CALL GRAPH	Set as Root, Reset Root, (?)	
HOT SPOTS		
SQL CALLS		
HTTP PARAMS		
COOKIES		
USER DATA		
ERROR DETAILS		
HARDWARE / MEM		
NODE PROBLEMS		
ADDITIONAL DATA		
	Name	Time (ms)
	StrutsActionProxy.execute	21 ms (self) 0.2 %
	StrutsAction.executeViewItems.getItems:44	6 ms (self) 0.1 %
	Proxy For Spring Bean - ItemServiceTarget.getItems	0 ms (self) 0 %
	Proxy For Spring Bean - ItemServiceTarget.invoke	0 ms (self) 0 %
	Spring Bean - ItemServiceTarget.getItems:16	0 ms (self) 0 %
	Spring Bean - ItemPersistence.getItems:33	0 ms (self) 0 %
	Spring Bean - oracleQueryExecutor.executeOracleQuery:22	9841 ms (self) 99.7 %

If there are multiple slow calls, repeat this step until you have resolved them all. However, there may be additional problems you haven't resolved. If any of the tier icons on the flow map show yellow or red, continue to [Step 3](#). Otherwise, you have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Business Transaction Dashboard](#)
- [Business Transaction Monitoring](#)
- [Business Transactions List](#)
- [Transaction Snapshots](#)

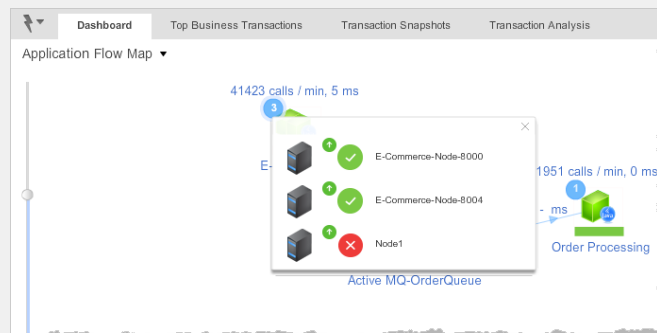
Step 3 - Affects 1 or more nodes?

Is the problem affecting all nodes in the slow tier?

▼ [How do I know?](#)

How do I know if the problem is affecting all nodes?

- In the Application or Tier Flow Map, click the number that represents how many nodes are in the tier. This provides a quick overview of the health of each node in the tier. The small circle icon indicates whether the server is up with the agent reporting, and the larger circle icon indicates Health Rule violation status.



If all the nodes are yellow or red, the answer to this question is Yes. Otherwise, the answer is No.

Yes – Go to [Step 4](#).

No – The problem is either in the node's hardware or in the way the software is configured on the node. (Because only one node is affected, the problem is probably not related to the code itself.)

In the left navigation pane, click **Servers -> App Servers -> <slow tier> -> <problematic node>** to display the Node Dashboard (flow map).



- Click the Dashboard tab to get a view of the overall health of the node.
- Click the Hardware tab; if the problem is hardware-related, contact your IT department.
- Click the Memory tab; sort on various column headings to determine if you need to add memory to the node, configure additional memory for the application, or take some other corrective action.

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Node Dashboard](#)

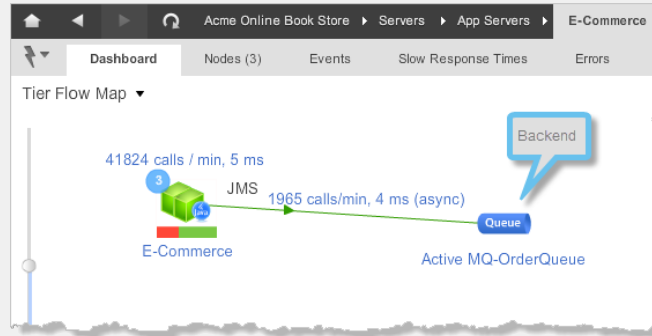
Step 4 - Backend problem?

Are the nodes in the slow tier linked to a backend (database or other remote service) that might be causing your problem?

How do I know?

How do I know if the nodes are linked to a backend (database or other remote service) that might be causing my problem?

- Display the Tier Flow Map. If any nodes are linked to a backend, links to those backends are displayed in the flow map.

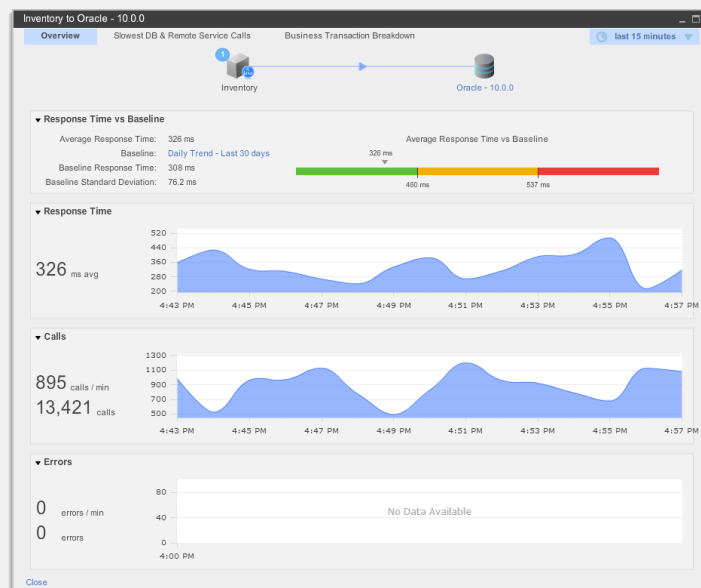


If a backend or the line connecting to a backend is yellow or red, the answer to this question is Yes. Otherwise, the answer is No.

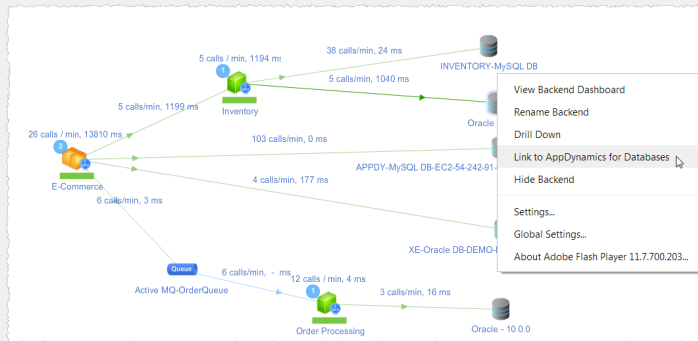
No – Go to [Step 5](#).

Yes –

- Click the line connecting to the backend to see an information window about the backend. (The contents of the information window vary depending on the type of backend.) Use the various tabs to find the source of the issue, or contact the team responsible for that backend.



If the backend is a database, right-click the database icon. You have a number of options that let you see the dashboard, drill down, etc. If you have AppDynamics for Databases, choose **Link to AppDynamics for Databases**. You can use AppDynamics for Databases to diagnose and resolve any backend issues, or work with your internal DBAs to troubleshoot the database, which is not instrumented in AppDynamics.



You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Backend Monitoring](#)
- [Configure Backend Detection for Java](#)
- [AppDynamics for Databases](#)

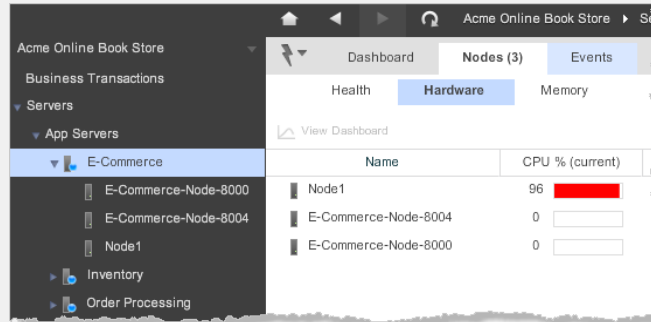
**Step 5 -
CPU
saturated?**

Is the CPU of the JVM saturated?

▼ [How do I know?](#)

How do I know if the CPU of the JVM is saturated?

1. Display the Tier Flow Map.
2. Click the Nodes tab, and then click the Hardware tab.
3. Sort by CPU % (current). **[Show me how.](#)**



If the CPU % is 90 or higher, the answer to this question is Yes. Otherwise, the answer is No.

Yes – Go to [Step 6](#).

No – The issue is probably related to a custom implementation your organization has developed. Take snapshots of the affected tier or node(s) and work with internal developers to resolve the issue.

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Monitor JVMs](#)

**Step 6 -
Significant
garbage
collection
activity?**

Is there significant garbage collection activity?

✓ [How do I know?](#)

How do I know if there is significant garbage collection activity?

- Display the Tier Flow Map.
- Click the Nodes tab, and then click the Memory tab.
- Sort by GC Time Spent to see how many milliseconds per minute is being spent on GC; 60,000 indicates 100%. **Show me how.**

Name	JVM % Heap	Max Heap	JVM CPU Burnt ...	GC Time Spent (ms/min)
E-Commerce-Node-8000	28.7	1712	59349	808
E-Commerce-Node-8004	5.6	1712	85	0
Node1	-	-	-	-

If GC Time Spent is higher than 500 ms, the answer to the question in Step 5 is Yes. Otherwise, the answer is No.

Yes – Go to [Step 7](#).

No – Go to [Step 8](#).

Need more help?

- [Memory Usage and Garbage Collection](#)

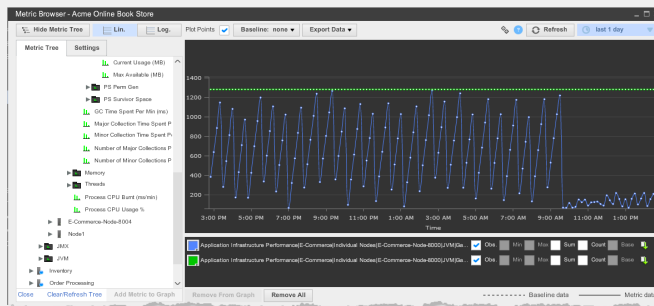
**Step 7 -
Memory
leak?**

Is there a memory leak?

✓ [How do I know?](#)

How do I know if there is a memory leak?

1. From the list of nodes displayed in the previous step (when you were checking for Garbage Collecting activity), double-click a node that is experiencing significant GC activity.
2. Click the Memory tab, then scroll down to display the Memory Pool graphs at the bottom of the window.
3. Double-click the PS Old Gen memory pools. **Show me how.**



If memory is not being released (use is trending upward), the answer to this question is Yes. Otherwise, the answer is No.

Yes – Use various AppDynamics features to track down the leak. One useful tool for diagnosing a memory leak is object instance tracking, which lets you track objects you are creating and determine why they aren't being released as needed. Using object instance tracking, you can pinpoint exactly where in the code the leak is occurring. For instructions on configuring object instance tracking, as well as links to other tools for finding and fixing memory leaks, see [Need more help?](#) below.

No – Increase the size of the JVM. If there is significant GC activity but there isn't a memory leak, then you probably aren't configuring a large enough heap size for the activities the code is performing. Increasing the available memory should resolve your problem.

Whether you answered Yes or No, you have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Troubleshoot Java Memory Leaks](#)
- [Troubleshoot Java Memory Thrash](#)
- [Configure and Use Object Instance Tracking for Java](#)

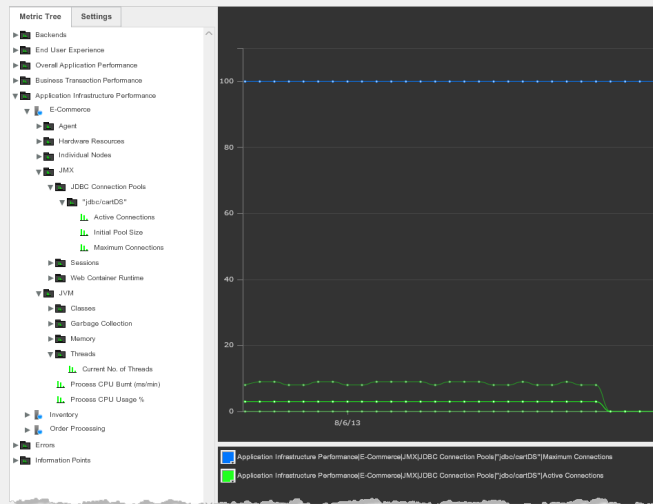
**Step 8 -
Resource
leak?**

Is there a resource leak?

✓ How do I know?

How do I know if there is a resource leak?

1. In the left Navigation pane, go to (for example) **Analyze -> Metric Browser -> Application Infrastructure Performance <slow tier> -> Individual Nodes -> <Problematic node> -> JMX -> JDBC Connection Pools -> <Pool name>**
2. Add the Active Connections and Maximum Connections metrics to the graph.
3. Repeat as needed for various pools your application is using.



If connections are not being released (use is trending upward), the answer to the question in Step 7 is Yes. Otherwise, the answer is No.

Yes – To determine where in your code resources are being created but not being released as needed, take a few thread dumps using standard commands on the problematic node. You can also create a diagnostic action within AppDynamics to create a thread dump; see [Thread Dump Actions](#).

No – Restart the JVM. If none of the above diagnostic steps addressed your issue, it's possible you're simply seeing a one-time unusual circumstance, which restarting the JVM can resolve.

Need more help?

- [Diagnostic Actions](#)

None of the above?

If slow response time persists even after you've completed the steps outlined above, you may need to perform deeper diagnostics.

If you can't find the information you need on how to do so in the AppDynamics documentation, consider posting a note about your problem in a community discussion topic. These discussions are monitored by customers, partners, and AppDynamics staff. Of course, you can also contact AppDynamics support.

Need more help?

- [AppDynamics Pro Documentation](#)
- [Community Discussion Boards](#) (If you don't see AppDynamics Pro as a topic, click Sign In at the upper right corner of the screen.)

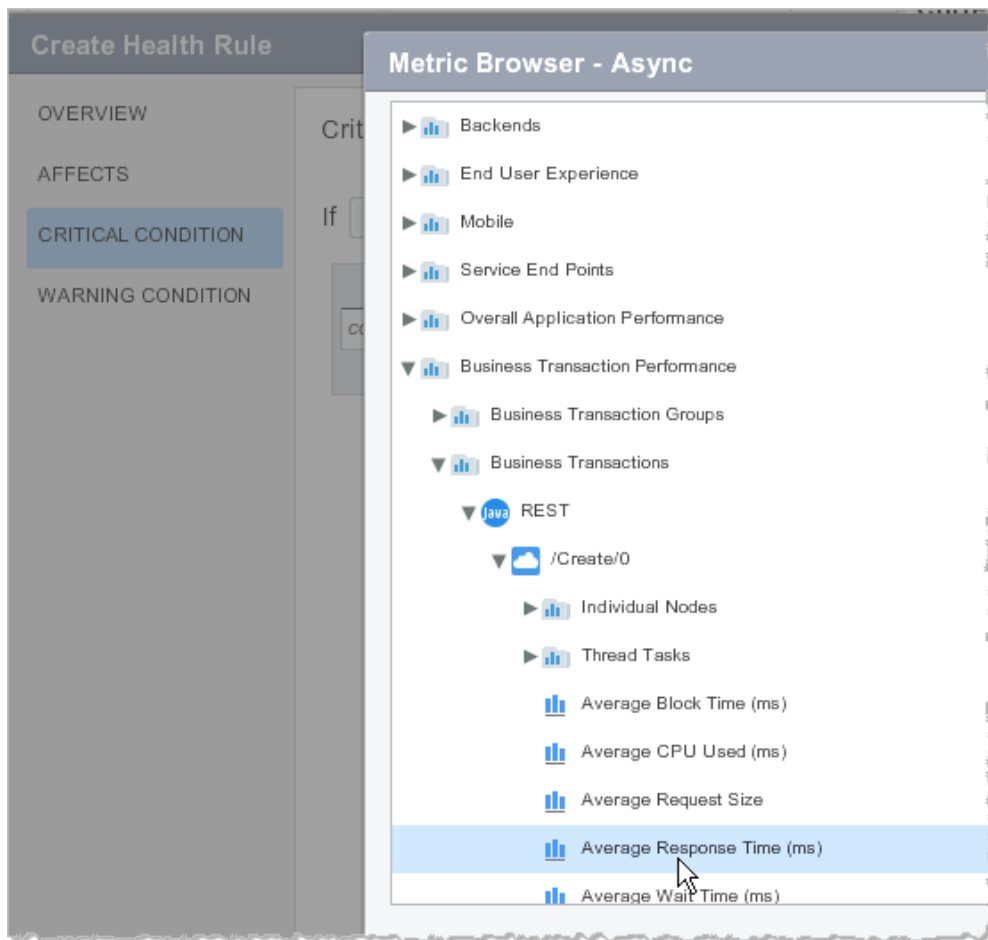
Configure Diagnostic Sessions For Asynchronous Activity

Automatic Diagnostic Sessions For Asynchronous Activity

Diagnostic sessions are triggered based on the performance metrics for a business transaction. The average response time of a business transaction does not include the execution time of its asynchronous activity. If you have asynchronous processing in your application, it might be possible for the originating transaction to execute within normal bounds even though the asynchronous activity takes longer than normal. To diagnose an issue like this, you can create a custom health rule based on the average response time (or other performance metric) of the asynchronous activity and use that health rule to set up a policy that triggers a diagnostic session on the transaction. The general steps to do this are described in the following example that uses a metric for an async thread task.

1. [Create a custom health rule](#) based on the asynchronous metric, such as average response time. The metrics for thread tasks are visible in the metric browser under the **Thread Tasks** node for transactions with asynchronous activity. Each thread task has an individual node

(usually its simple class name). Remember to select Custom as the type for the health rule.



2. [Create a policy](#) that is based on the baseline of the asynchronous metric of interest, for example, the average response time.
3. Configure the policy to trigger a diagnostic session on the affected business transaction.

Troubleshoot Java Memory Issues

Troubleshoot Java Memory Leaks

- [Memory Leaks in a Java Environment](#)
- [AppDynamics Java Automatic Leak Detection](#)
 - [Automatic Leak Detection Support](#)
 - [Conditions for Troubleshooting Java Memory Leaks](#)
- [Workflow to Troubleshoot Memory Leaks](#)
 - [Monitor Memory for Potential JVM Leaks](#)
 - [Enable Memory Leak Detection](#)
 - [Troubleshoot Memory Leaks](#)
 - [Select the Collection Object to Monitor](#)
 - [Use Content Inspection](#)
 - [Use Access Tracking](#)
- [Learn More](#)

Memory Leaks in a Java Environment

While the JVM's garbage collection greatly reduces the opportunities for memory leaks to be introduced into a codebase, it does not eliminate them completely. For example, consider a web page whose code adds the current user object to a static set. In this case, the size of the set grows over time and could eventually use up significant amounts of memory. In general, leaks occur when an application code puts objects in a static collection and does not remove them even when they are no longer needed.

In high workload production environments if the collection is frequently updated, it may cause the applications to crash due to insufficient memory. It could also result in system performance degradation as the operating system starts paging memory to disk.

AppDynamics Java Automatic Leak Detection

AppDynamics automatically tracks every Java collection (for example, HashMap and ArrayList) that meets a set of criteria defined below. The collection size is tracked and a linear regression model identifies whether the collection is potentially leaking. You can then identify the root cause of the leak by tracking frequent accesses of the collection over a period of time.

Once a collection is qualified, its size, or number of elements, is monitored for long term growth trend. A positive growth indicates that the collection is potentially leaking!

Once a leaking collection is identified, the agent automatically triggers diagnostics every 30 minutes to capture a shallow content dump and activity traces of the code path and business transactions that are accessing the collection. By drilling down into any leaking collection monitored by the agent, you can manually trigger Content Summary Capture and Access Tracking sessions. See [Configure Automatic Leak Detection for Java](#)

You can also monitor memory leaks for custom memory structures. Typically custom memory structures are used as caching solutions. In a distributed environment, caching can easily become a prime source of memory leaks. It is therefore important to manage and track memory statistics for these memory structures. To do this, you must first configure custom memory structures. See [Configure and Use Custom Memory Structures for Java](#).

Automatic Leak Detection Support

Ensure AppDynamics supports Automatic Leak Detection on your JVM. See [JVM Support](#).

Conditions for Troubleshooting Java Memory Leaks

Automatic Leak Detection uses On Demand Capture Sessions to capture any actively used collections (i.e. any class that implements JDK Map or Collection interface) during the Capture period (default is 10 minutes) and then qualifies them based on the following criteria:

For a collection object to be identified and monitored, it must meet the following conditions:

- The collection has been alive for at least N minutes. Default is 30 minutes, configurable with the [minimum-age-for-evaluation-in-minutes](#) node property.
- The collection has at least N elements. Default is 1000 elements, configurable with the [minimum-number-of-elements-in-collection-to-deep-size](#) node property.
- The collection Deep Size is at least N MB. Default is 5 MB, configurable with the [minimum-size-for-evaluation-in-mb](#) property.

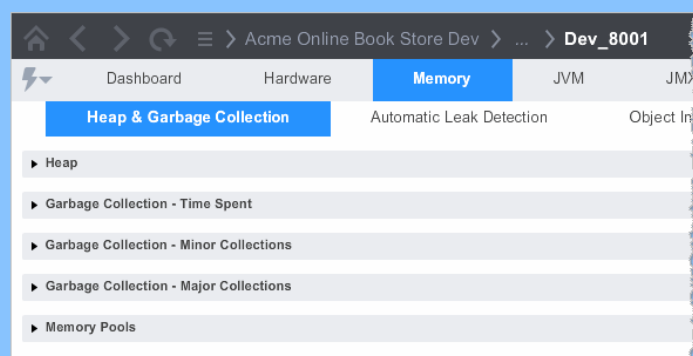
The Deep Size is calculated by traversing recursive object graphs of all the objects in the collection.

See [App Agent Node Properties](#) and [App Agent Node Properties Reference by Type](#).

Workflow to Troubleshoot Memory Leaks

Use the following workflow to troubleshoot memory leaks on JVMs that have been identified with a potential memory leak problem:

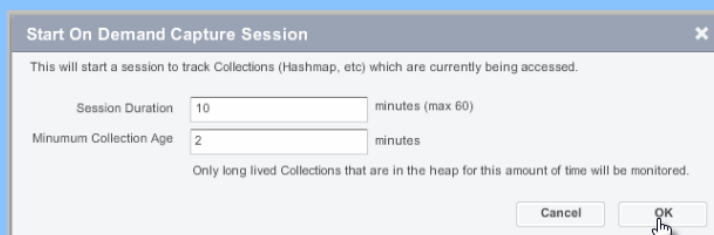
Monitor JVM Memory for potential memory leak



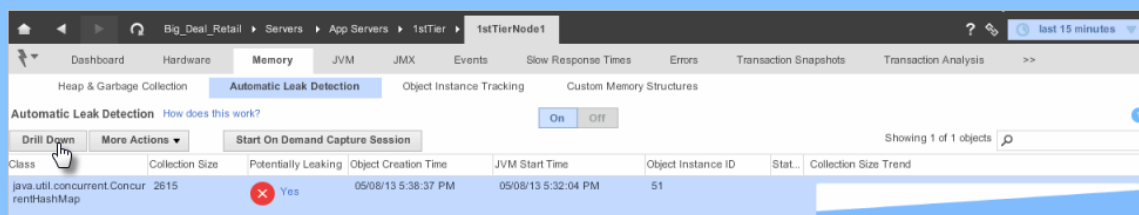
Enable Automatic Leak Detection



Start On Demand Capture Session

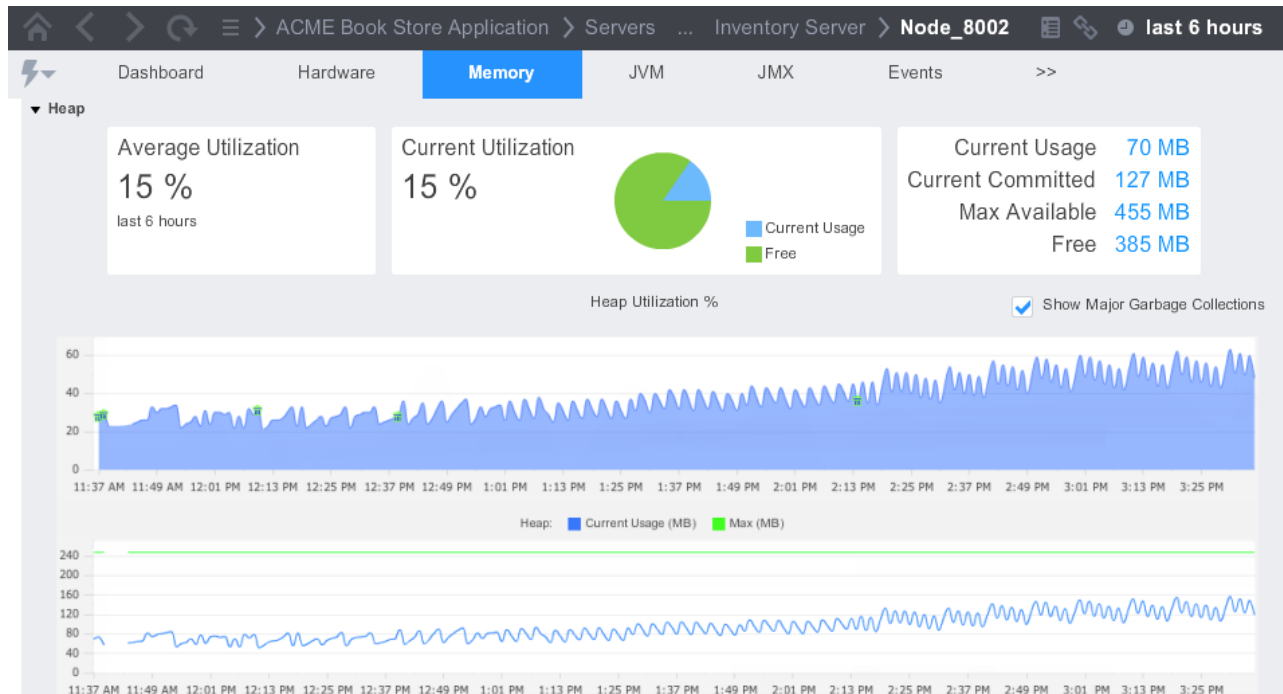


Detect and troubleshoot leaking condition

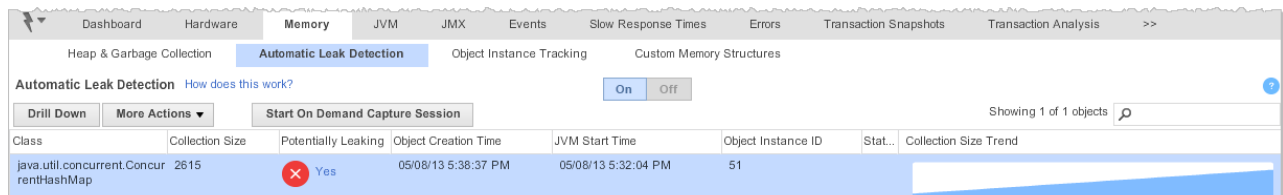


Monitor Memory for Potential JVM Leaks

Use the Node Dashboard to identify the memory leak. A possible memory leak is indicated by a growing trend in the heap as well as the old/tenured generation memory pool.



An object is automatically marked as a potentially leaking object when it shows a positive and steep growth slope.



The Automatic Memory Leak dashboard shows:

- Collection Size: The number of elements in a collection.
- Potentially Leaking: Potentially leaking collections are marked as red. You should 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.

of Elements in a Collection



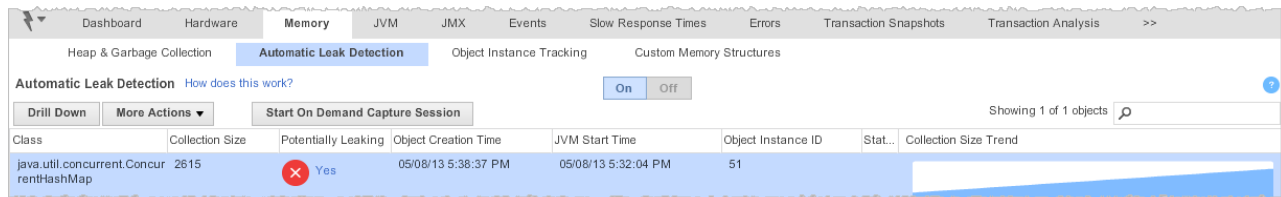
Tip: To identify long-lived collections compare the JVM start time and Object Creation Time.

If you cannot see any captured collections, ensure that you have correct configuration for detecting potential memory leaks.

Enable Memory Leak Detection

Before enabling memory leak detection, identify the potential JVMs that may have a leak. See [Detect Memory Leaks](#).

Memory leak detection is available through the Automatic Leak Detection feature. Once the Automatic Leak Detection feature is turned on and a capture session has been started, AppDynamics tracks all frequently used collections; therefore, using this mode results in a higher overhead. Turn on Automatic Leak Detection mode only when a memory leak problem is identified and then start an On Demand Capture Session to start monitoring frequently used collections and detect leaking collections.



Turn this mode off after you have identified and resolved the leak.

To achieve optimum performance, start diagnosis on an individual collection at a time.

Troubleshoot Memory Leaks

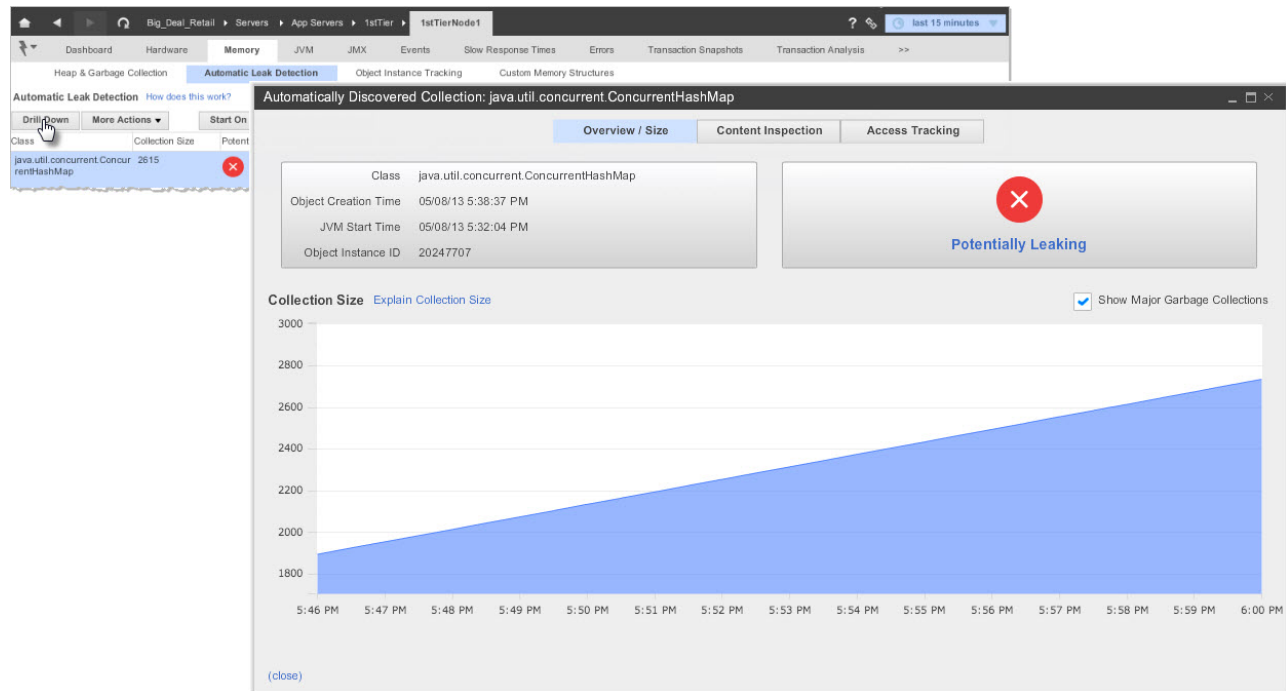
After detecting a potential memory leak, troubleshooting the leak involves performing the following three actions:

- [Select the Collection Object that you want to monitor](#)
- [Use Content Inspection](#)
- [Use Access Tracking](#)

Select the Collection Object to Monitor

1. On the Automatic Leak Detection dashboard, select the name of the class that you want to monitor.
2. Click **Drill Down** on the top left-hand side of the memory leak dashboard. Alternatively right-click the class name and click **Drill Down**.

IMPORTANT: To achieve optimum performance, start the troubleshooting session on a single collection object at a time.



Use Content Inspection

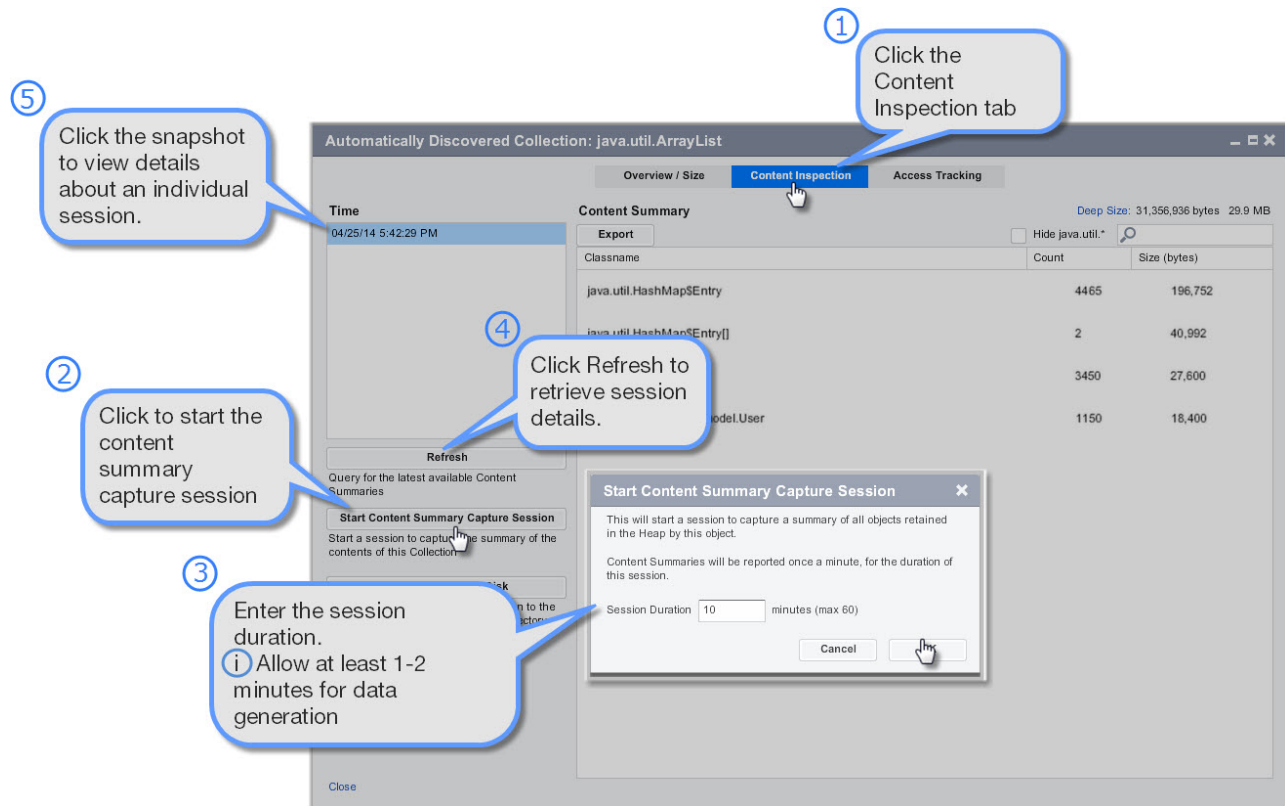
Use Content Inspection to identify which part of the application the collection belongs to so that you can start troubleshooting. It allows monitoring histograms of all the elements in a particular collection.

As described above in [Workflow to Troubleshoot Memory Leaks](#), enable Automatic Leak Detection, start an On Demand Capture Session, select the object you want to troubleshoot, and then follow the steps listed below:

1. Click the Content Inspection tab.
2. Click **Start Content Summary Capture Session** to start the content inspection session.
3. Enter the session duration. Allow at least 1-2 minutes for data generation.
4. Click **Refresh** to retrieve the session data.
5. Click on the snapshot to view details about an individual session.

i Exporting Troubleshooting Information

You can also export the troubleshooting information into Excel files using the Export button under Content Summary.



Use Access Tracking

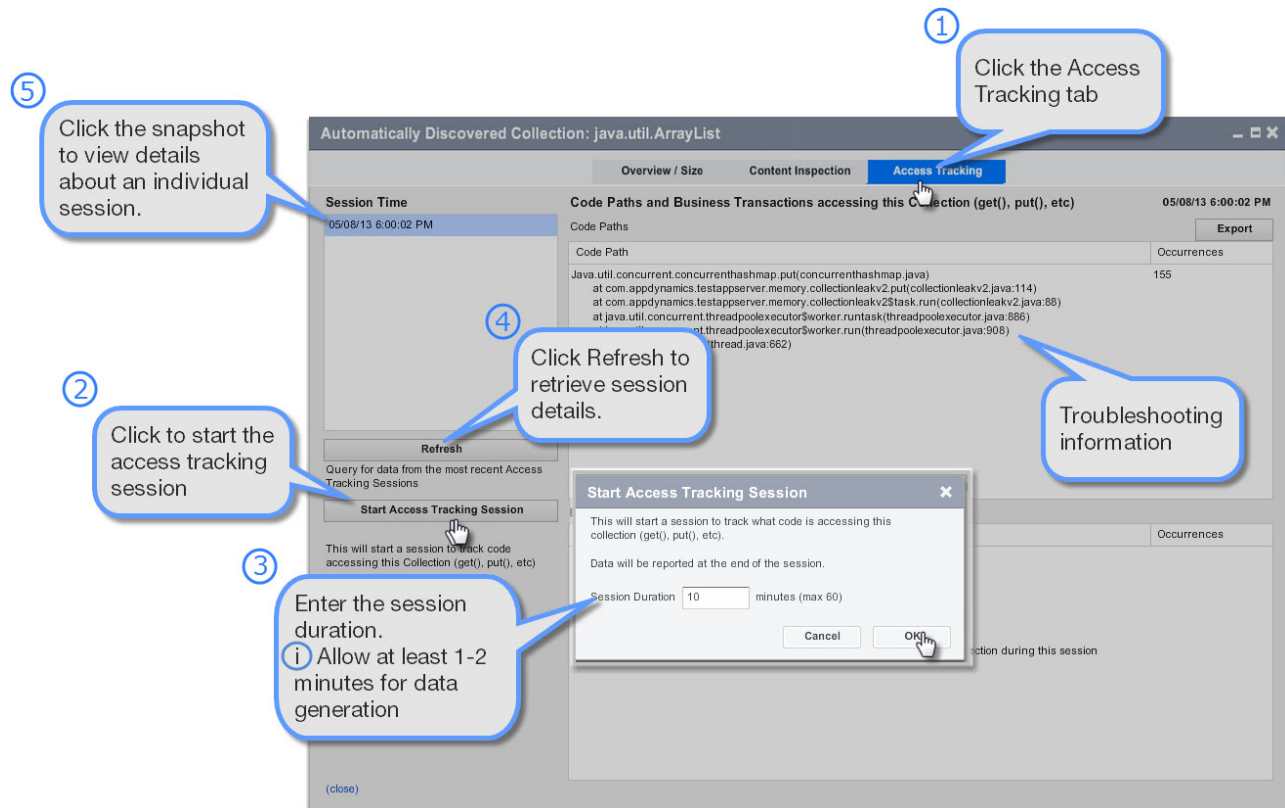
Use Access Tracking to view the actual code paths and business transactions accessing the collections object.

As described above in [Workflow to Troubleshoot Memory Leaks](#), enable Automatic Leak Detection, start an On Demand Capture Session, select the object you want to troubleshoot, and then follow the steps listed below:

1. Select the Access Tracking tab
2. Click **Start Access Tracking Session** to start the tracking session.
3. Enter the session duration. Allow at least 1-2 minutes for data generation.
4. Click **Refresh** to retrieve session data.
5. Click on the snapshot to view details about an individual session.

i Exporting Troubleshooting Information

You can also export the troubleshooting information into Excel files using the Export button under Content Summary.



Learn More

- [App Agent Node Properties](#)
- [Monitor JVMs](#)
- [Metric Browser](#)

Troubleshoot Java Memory Thrash

- [Memory Thrash and Object Instance Tracking](#)
- [Prerequisites for Object Instance Tracking](#)
 - [Specifying the Classpath](#)
- [Workflow for Detecting and Troubleshooting Memory Thrash](#)
- [Analyzing Memory Thrash](#)
 - [To analyze memory thrash problems](#)
 - [To verify memory thrash](#)
 - [Troubleshooting Java Memory Thrash Using Allocation Tracking](#)
 - [To use allocation tracking](#)

Memory Thrash and Object Instance Tracking

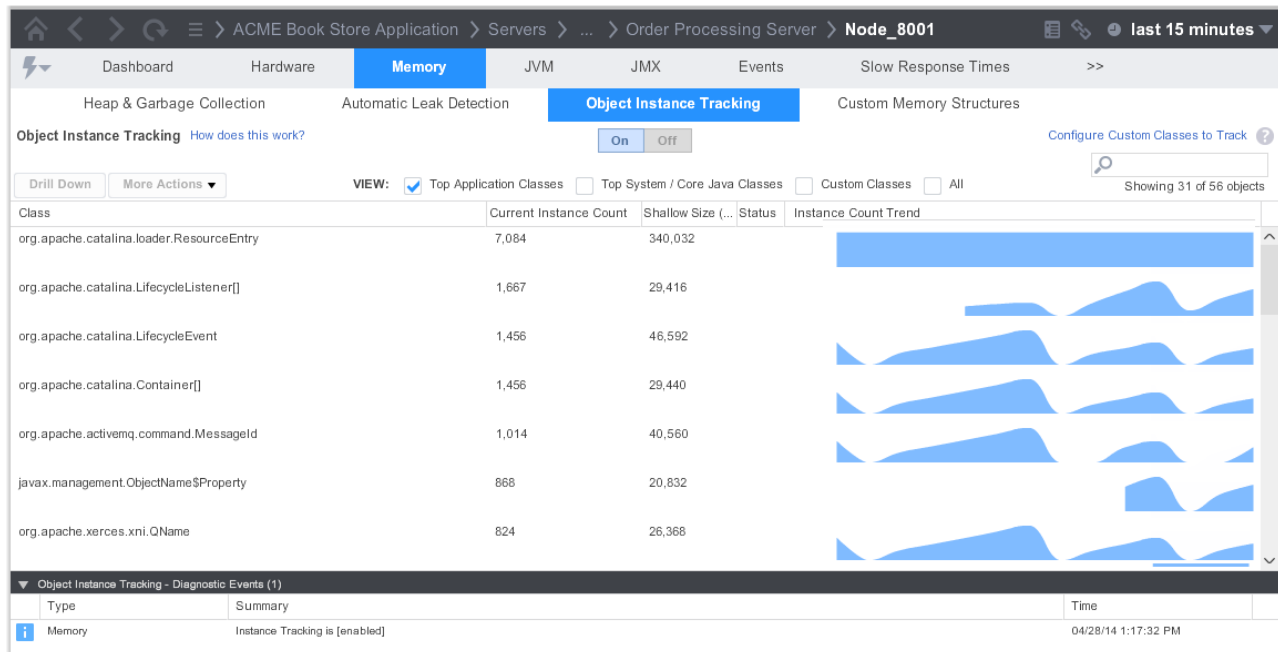
Memory thrash is caused when a large number of temporary objects are created in very short intervals. Although these objects are temporary and are eventually cleaned up, the garbage collection mechanism may struggle to keep up with the rate of object creation. This may cause application performance problems. Monitoring the time spent in garbage collection can provide insight into performance issues, including memory thrash. For example, an increase in the

number of spikes for major collections either slows down a JVM or indicates potential memory thrash. Use object instance tracking to isolate the root cause of the memory thrash. To configure and enable object instance tracking, see [Configure and Use Object Instance Tracking for Java](#).

AppDynamics automatically tracks the following classes:

- Application Classes
- System Classes

The Object Instance Tracking feature maps a histogram of every object in the JVM. The Object Instance Tracking dashboard not only provides the number of instances for a particular class but also provides the shallow memory size (the memory footprint of the object and the primitives it contains) used by all the instances.



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 probably be setup correctly, but if you are running with the JRE you must add tools.jar to JRE_HOME/lib/ext and restart the JVM for this feature to start working. You can find the tools.jar file in JAVA_HOME/lib/tools.jar.
- In some cases you might also need to copy libattach.so (Linux) or attach.dll (Windows) from your JDK to your JRE.
- Depending on the JDK version, you may also need to specify the classpath as shown below (along with other -jar options).

Specifying the Classpath

When using a JDK tool, set the classpath using the -classpath option. This sets the classpath for the application only. For example:

On Windows

```
java -classpath <complete-path-to-tools.jar>;%CLASSPATH% -jar myApp.jar
```

OR

On Unix

```
java -classpath <complete-path-to-tools.jar>:$CLASSPATH -jar myApp.jar
```

Alternatively, you can set the CLASSPATH variable for your entire environment. For example:

On Windows


```
SET CLASSPATH=%CLASSPATH%;%JAVA_HOME%\lib\tools.jar
```

On Unix

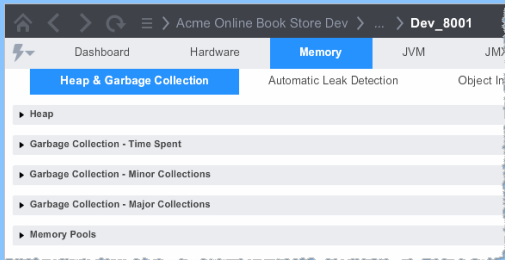
```
CLASSPATH=$CLASSPATH:$JAVA_HOME/lib/tools.jar
```

Workflow for Detecting and Troubleshooting Memory Thrash

The following diagram outlines the workflow for monitoring and troubleshooting memory thrash problems in a production environment.

 To monitor memory leaks, on the node dashboard, use the Memory -> Automatic Leak Detection subtab. See [Troubleshoot Java Memory Leaks](#).

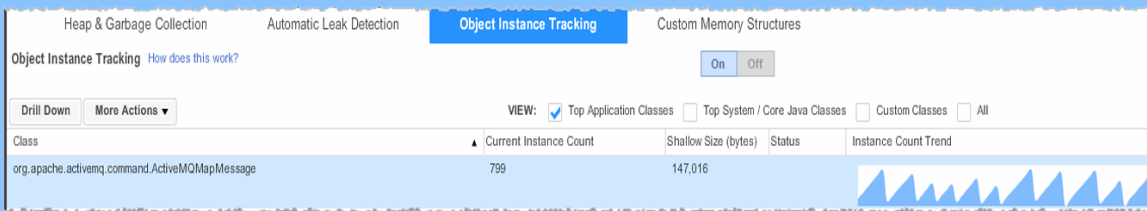
Monitor JVM Memory



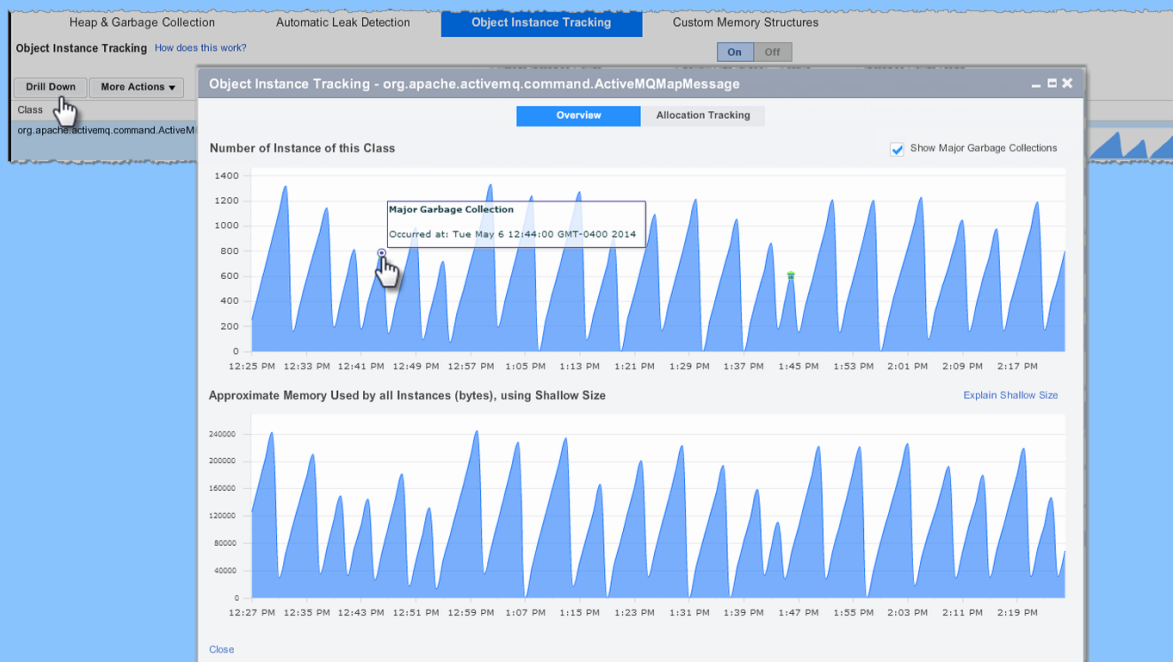
Enable Object Instance Tracking



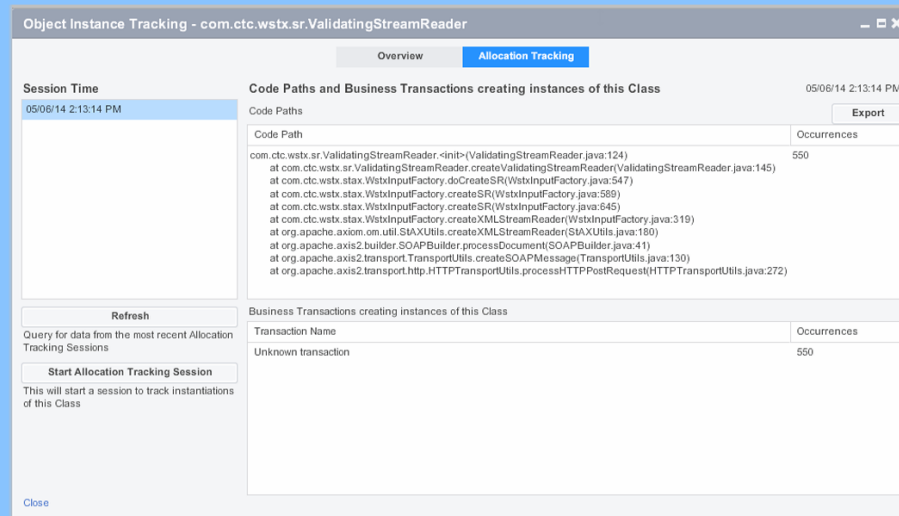
Detect Memory Thrash Problem



Verify Memory Thrash Problem



Troubleshoot Memory Thrash Problem



Analyzing Memory Thrash


To analyze memory thrash problems

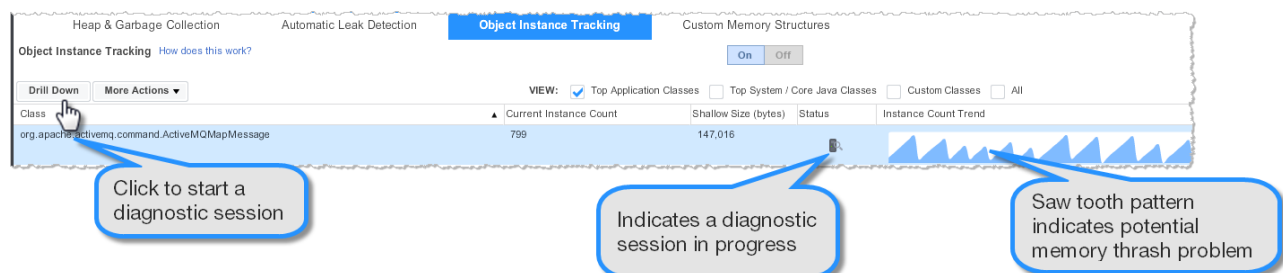
Once a memory thrash problem is identified in a particular collection, start the diagnostic session by drilling down into the suspected problematic class.

1. Select the class name to monitor and click **Drill Down** at the top of the Object Instance Tracking dashboard.

Or right click the class name and select the Drill Down option.

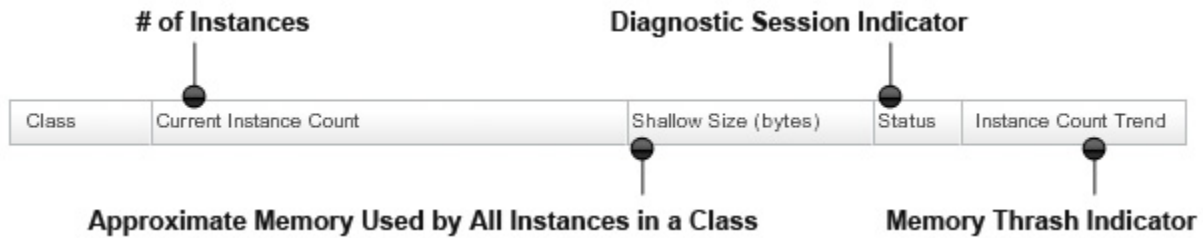
i For optimal performance, trigger a drill down action on a single instance or class name at a time.

After the drill down action is triggered, data collection for object instances is performed every minute. This data collection is considered to be a diagnostic session and the Object Instance Tracking dashboard for that class is updated with this icon , to indicate that a diagnostic session is in progress.



The Object Instance Tracking dashboard indicates possible cases of memory thrash.

The following provides more detail on the meaning of each of the columns on the Object Instance Tracking dashboard.



Prime indicators of memory thrash problems are:

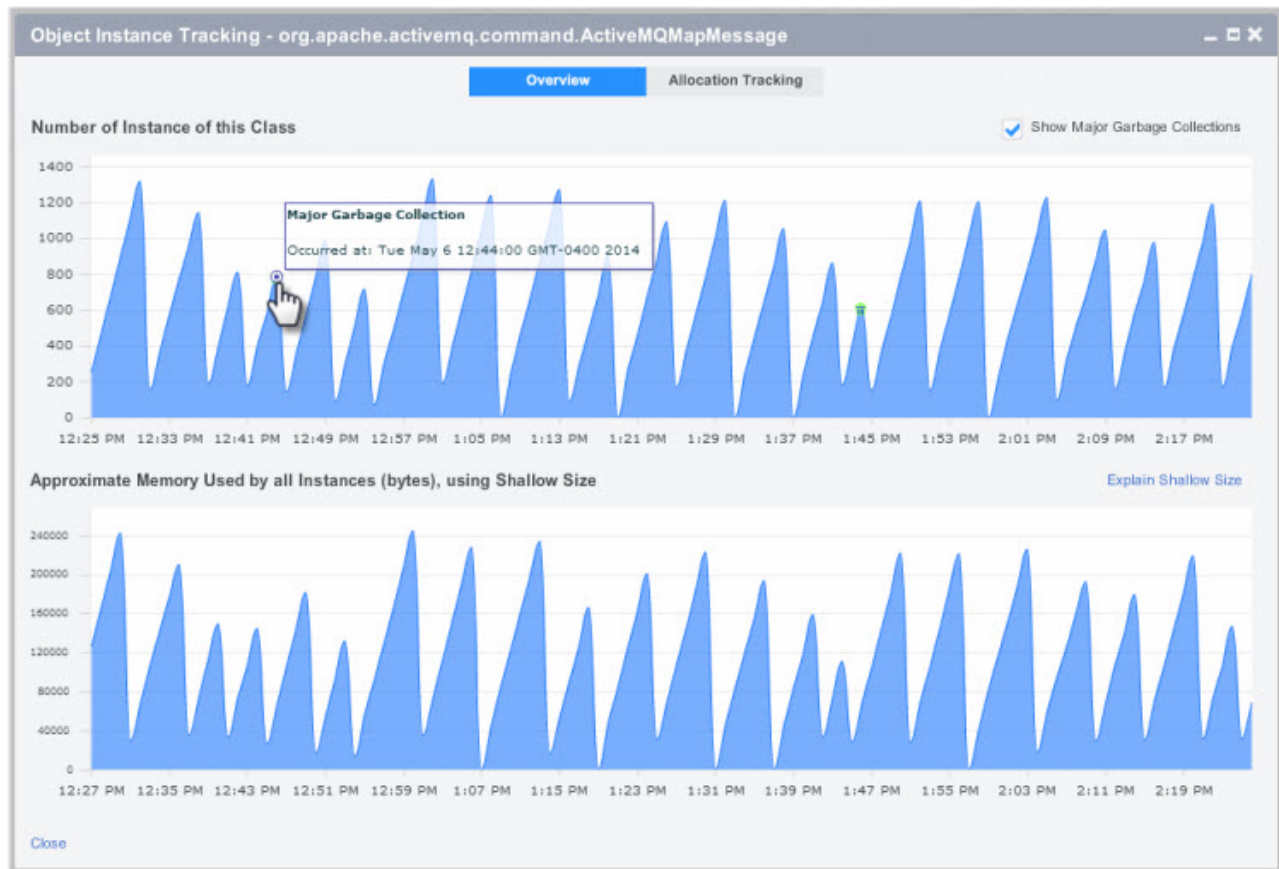
- **Current Instance Count:** A high number indicates possible allocation of large number of temporary objects.
- **Shallow Size:** A large number for shallow size signals potential memory thrash.
- **Instance Count Trend:** A saw wave is an instant indication of memory thrash.

If you suspect you have a memory thrash problem at this point, then you should verify that this is the case. See [To verify memory thrash](#).

To verify memory thrash

1. Select the class name to monitor and click **Drill Down** at the top of the Object Instance Tracking dashboard.
2. On the Object Instance Tracking window, click **Show Major Garbage Collections**.

The following Object Instance Tracking Overview provides further evidence of a memory thrash problem.



If the instance count doesn't vary with the garbage collection cycle, it is an indication of potential leak and not a memory thrash problem. See [Troubleshoot Java Memory Leaks](#).

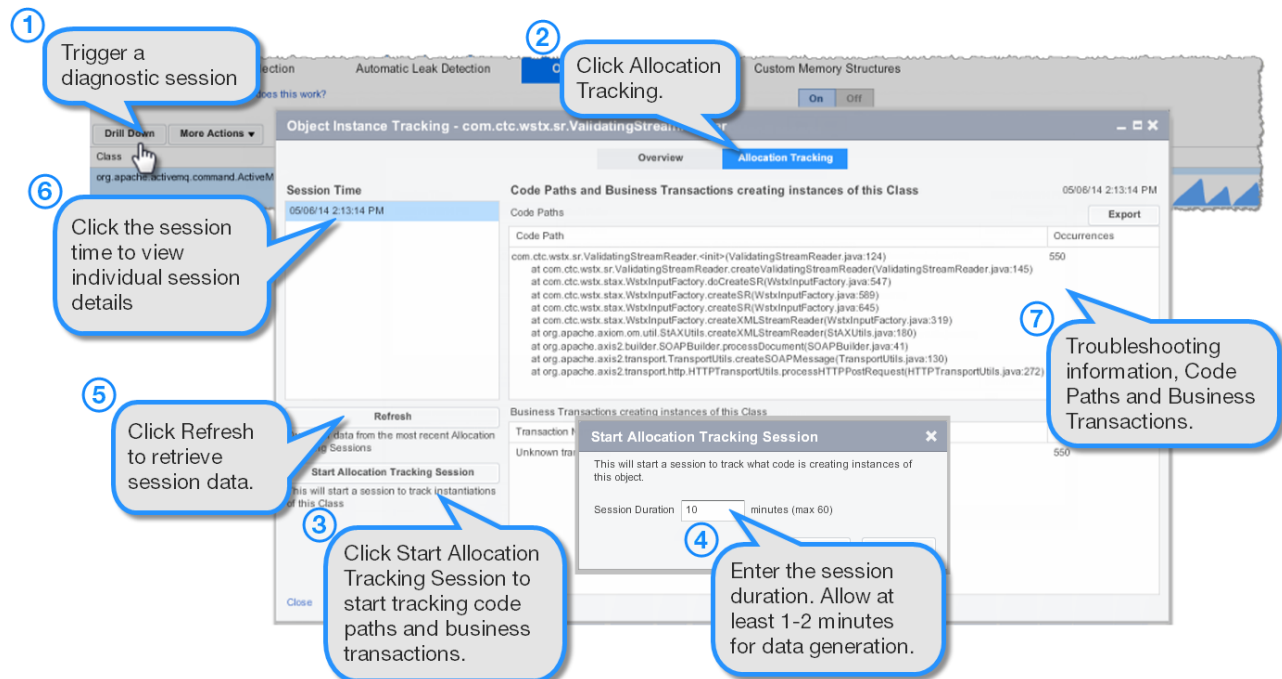
Troubleshooting Java Memory Thrash Using Allocation Tracking

Allocation Tracking tracks all the code paths and those business transactions that are allocating instances of a particular class.

Allocation tracking detects those code path/business transactions that are creating and throwing away instances.

To use allocation tracking

1. Using the Drill Down option, trigger a diagnostic session.
2. Click the Allocation Tracking tab.
3. Click **Start Allocation Tracking Session** to start tracking code paths and business transactions.
4. Enter the session duration and allow at least 1-2 minutes for data generation.
5. Click **Refresh** to retrieve the session data.
6. Click on a session to view its details.
7. Use the Information presented in the Code Paths and Business Transaction panels to identify the origin of the memory thrash problem.



Detect Code Deadlocks for Java

- [Code Deadlocks and their Causes](#)
 - [Finding Deadlocks using the Events List](#)
 - [To Examine a Code Deadlock](#)
 - [Finding Deadlocks Using the REST API](#)
- [Learn More](#)

i Read a real-life story about how AppDynamics helped identify code deadlocks and reduce the risk to revenue!

By default the agent detects code deadlocks. You can find deadlocks and see their details using the Events list or the REST API.

Code Deadlocks and their Causes

In a multi-threading development environment, it is common to use more than a single lock. However sometimes deadlocks will occur. Here are some possible causes:

- The order of the locks is not optimal
- The context in which they are being called (for example, from within a callback) is not correct
- Two threads may wait for each other to signal an event

Finding Deadlocks using the Events List

Select **Code Problems** (or just **Code Deadlock**) in the Filter By Event Type list to see code deadlocks in the Events list. See [Filter and Analyze Events](#). The following list shows two

deadlocks in the ECommerce tier.

The screenshot shows the AppDynamics Events console. On the left, under 'FILTER BY EVENT TYPE', 'Policy Violations' and 'Code Problems' are expanded. Under 'Code Problems', 'Code Deadlock' and 'Resource Pool Limit Reached' are checked. The main table shows two 'Code Deadlock' events. The first event is selected, and its details are shown on the right.

Type	Summary	Time	Business T	Tier	Node	Policy
Code Deadlock	JVM de...	09/19/12 4:		EComm...		
Code Deadlock	JVM de...	09/19/12 4:		EComm...		

To Examine a Code Deadlock

1. Double-click the deadlock event in the events list.
The Code Deadlock **Summary** tab displays.

The screenshot shows the 'Summary' tab for a Code Deadlock event. The event is critical, occurred on 09/19/12 at 4:16:49 PM, and the summary is 'JVM deadlock detected'. It occurred on the 'ECommerce Server' tier, specifically on 'Node_8000'.

Property	Value
Severity	Critical
Type	Code Deadlock
Time	09/19/12 4:16:49 PM
Summary	JVM deadlock detected
Tier	ECommerce Server
Node	Node_8000

2. To see details about the deadlock click the **Details** tab and scroll down.

The screenshot shows the 'Details' tab for a Code Deadlock event. It displays the thread stack for three threads involved in the deadlock. Thread 11 is holding a lock that Thread 12 is waiting for, and Thread 13 is holding a lock that Thread 11 is waiting for.

```

t1
  Name[t1]Thread ID[968]
  Deadlocked on Lock[java.lang.String@52be1266] held by thread [t2] Thread ID[969]
  Thread stack [
    com.appdynamics.DeadlockingThread.g(DeadlockingThread.java:34)
    com.appdynamics.DeadlockingThread.f(DeadlockingThread.java:28)
    com.appdynamics.DeadlockingThread.run(DeadlockingThread.java:20)
  ]

t2
  Name[t2]Thread ID[969]
  Deadlocked on Lock[java.lang.String@4140ac33] held by thread [t3] Thread ID[970]
  Thread stack [
    com.appdynamics.DeadlockingThread.g(DeadlockingThread.java:34)
    com.appdynamics.DeadlockingThread.f(DeadlockingThread.java:28)
    com.appdynamics.DeadlockingThread.run(DeadlockingThread.java:20)
  ]

t3
  Name[t3]Thread ID[970]
  Deadlocked on Lock[java.lang.String@4d17a75f] held by thread [t1] Thread ID[968]
  Thread stack [
  
```

Finding Deadlocks Using the REST API

You can detect a DEADLOCK event-type using the AppDynamics REST API. For details see the example [Retrieve event data](#).

Learn More

- [Use the AppDynamics REST API](#)

Troubleshooting Tutorials for Java

Tutorial for Java - Business Transaction Health Drilldown

Business Transaction Drilldown



Download MP4 version: [BTHealthDrilldown.mp4](#)

Download QuickTime version: [BTHealthDrilldown.mov](#)

Tutorial for Java - Exceptions

- [The Exceptions](#)
- [Drill Down into the HTTP Error Code Exception](#)
- [Drill Down into the AxisFault Exception](#)
- [Drill Down into the Logger Exception](#)
- [See How Exceptions are Configured](#)
- [Learn More](#)

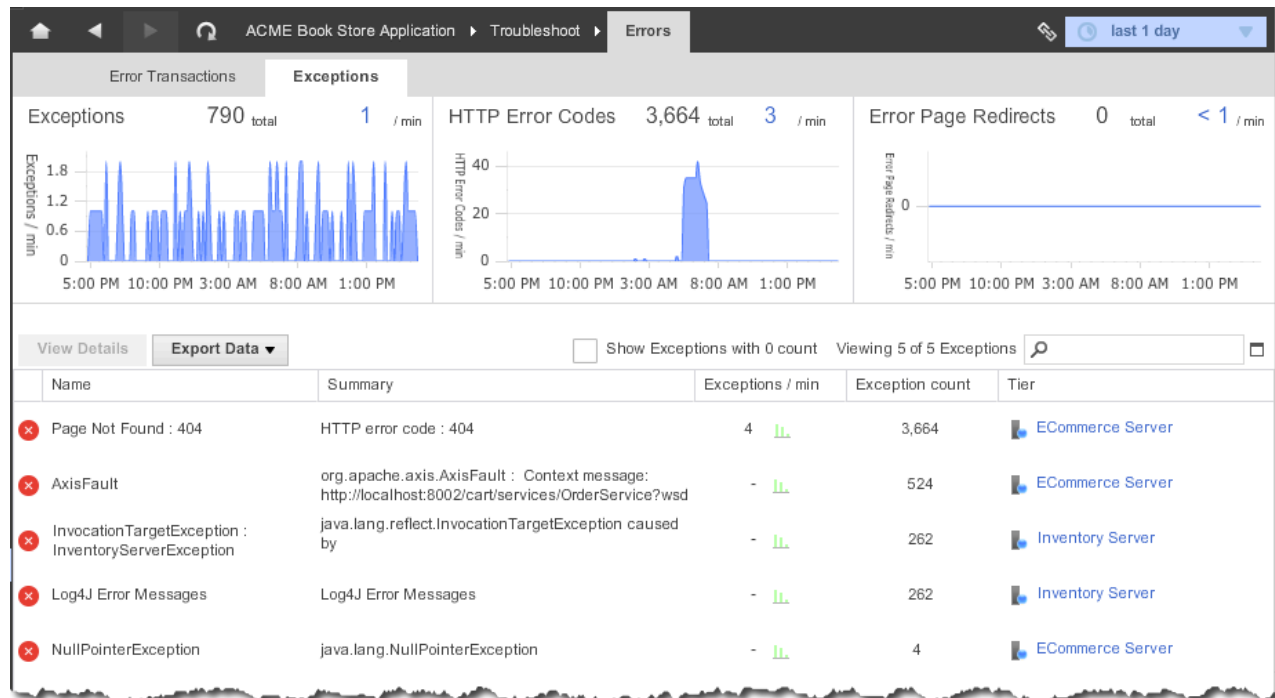
The Exceptions

An exception is a code-logged message outside the context of a business transaction. Common exceptions include code exceptions or logged errors, HTTP error codes, and error page redirects.

Exceptions display in the Exceptions pane of many dashboards.

Exceptions		
Exceptions	790 total	1 / min
HTTP Error Codes	3,664 total	3 / min
Error Page Redirects	0 total	< 1 / min

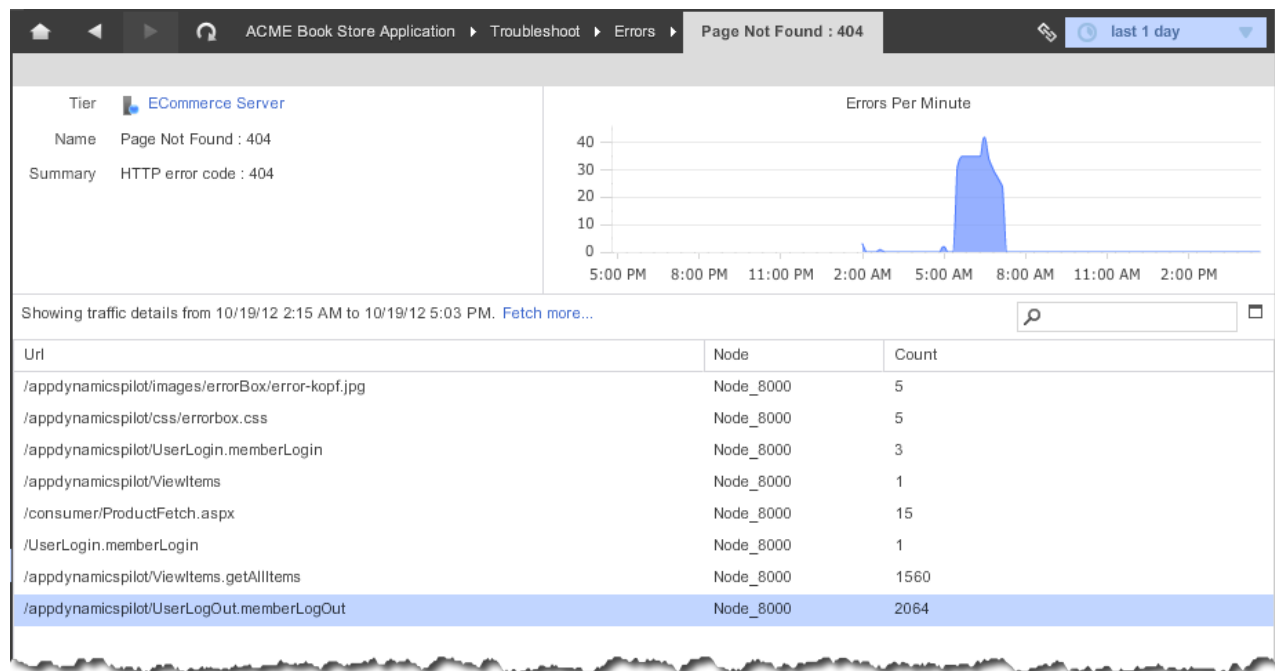
Click Exceptions to quickly see a list, ordered by frequency.



Drill Down into the HTTP Error Code Exception

Notice the spike in the HTTP Error Codes graph, and that the "Page Not Found: 404 error" is the most frequent.

To find out more about the 404 error, click the row.

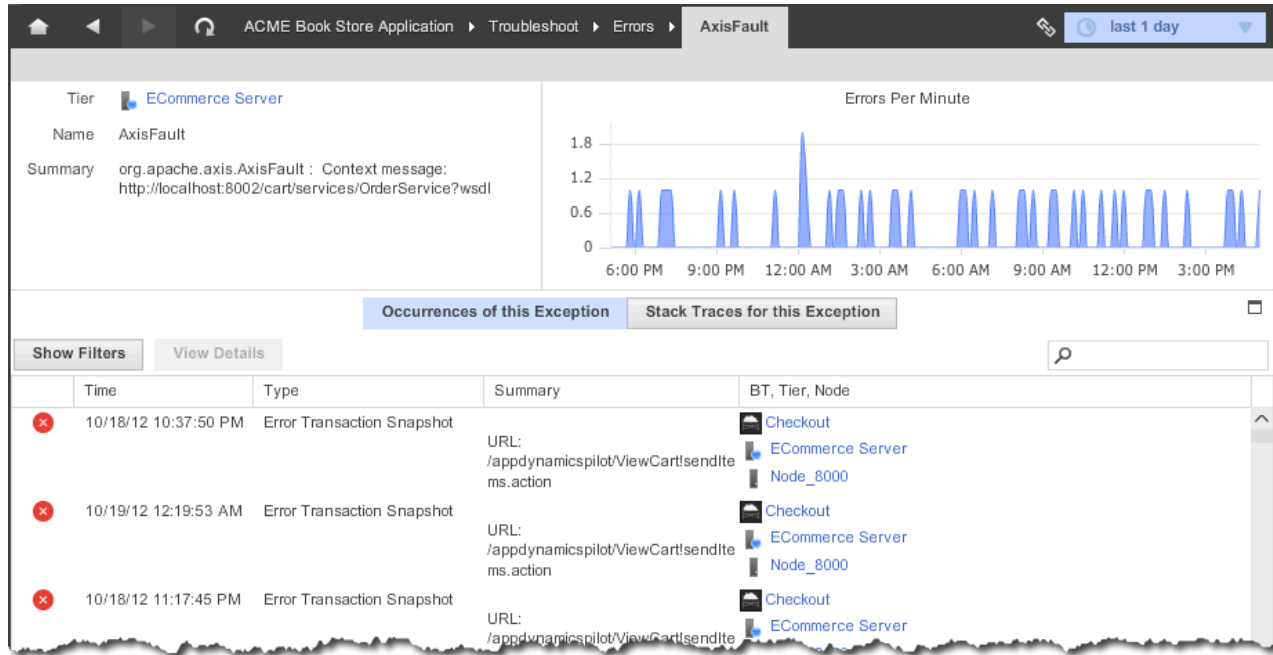


The list of URLs shows pages that have 404 errors. The memberLogout and getAllItems URLs have the most 404 errors. You can provide this information to the web team to determine why

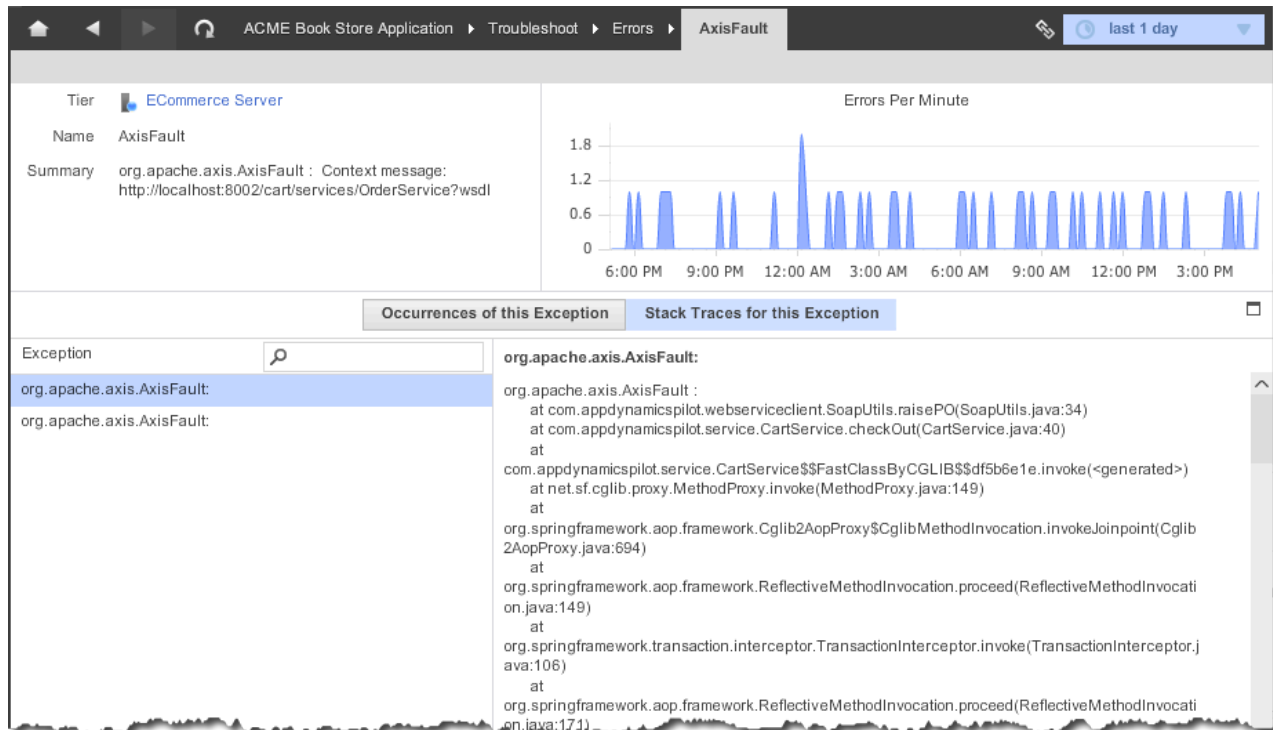
those pages have so many 404 errors.

Drill Down into the AxisFault Exception

In the Exceptions tab, click the AxisFault row. A list of error snapshots shows the affected URL, tier, and node.



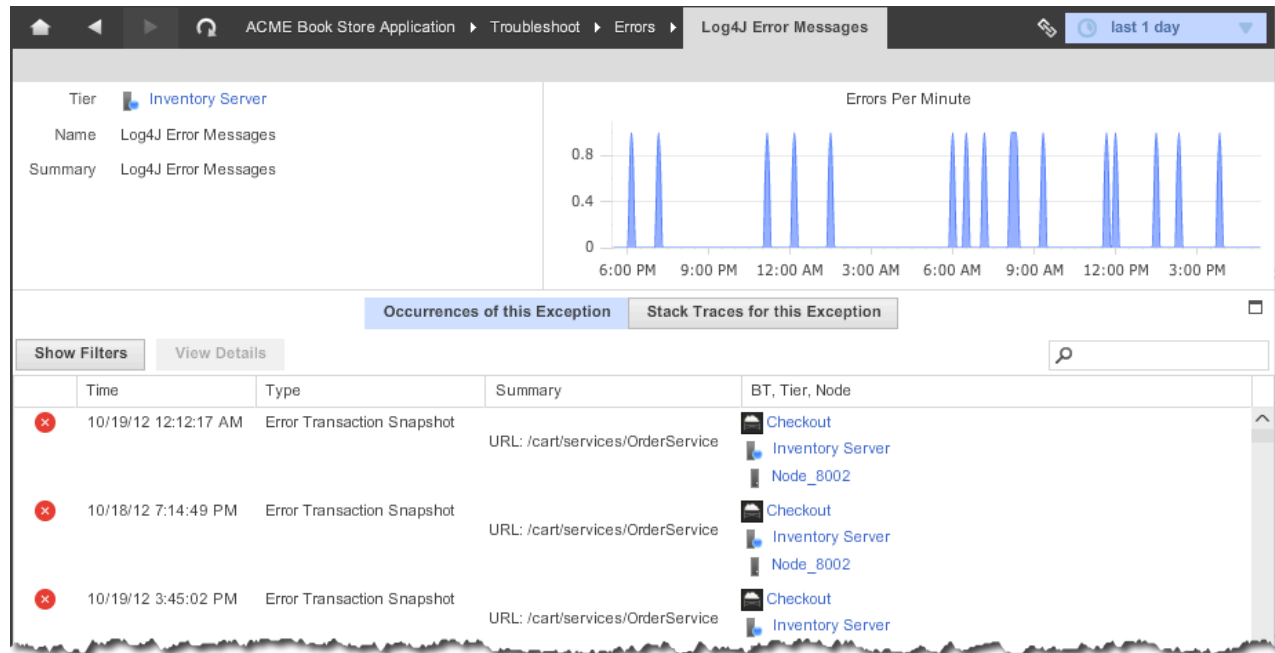
Click a row and then click the Stack Traces for This Exception tab to drill down further. Then click on one of the exceptions to see the stack trace.



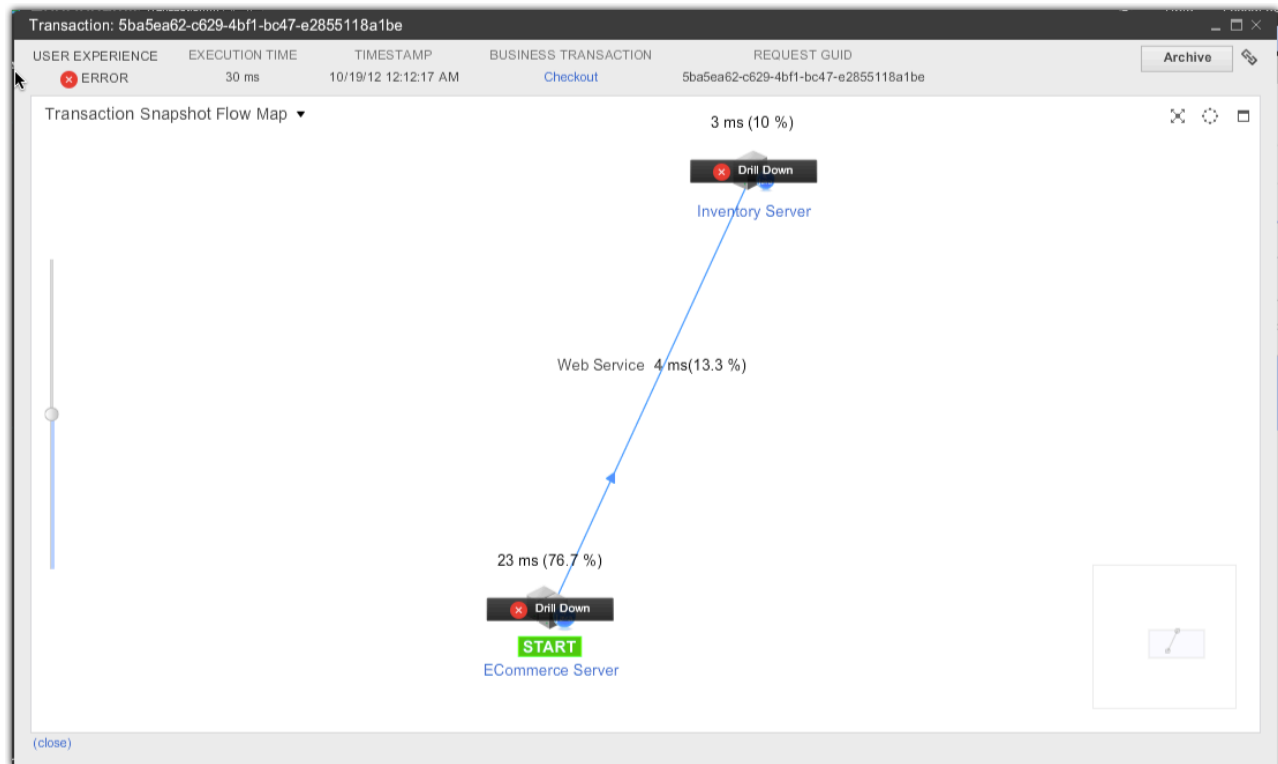
Share the stack trace with the development team to solve the problem.

Drill Down into the Logger Exception

In the Exceptions tab, click the Log4J Error Messages row. A list of error transaction snapshots shows the affected URL, business transaction, tier, and node. You can see a graph of the errors-per-minute data.



Click on a row to see the flow map for the error transaction snapshot.



The icons for both tiers have a Drill Down button. Click the Drill Down button on Ecommerce tier; it also says "Start", indicating that the transaction started on this tier.

The Call Drill Down shows the summary of the error message.

Call Drill Down. Exe Time: 30 ms Timestamp: 10/19/12 12:12:17 AM BT: Checkout GUID: 5ba5ea62-c629-4bf1-bc47-e2855118a1be

SUMMARY

SQL CALLS

HTTP PARAMS

COOKIES

USER DATA

ERROR DETAILS

HARDWARE / MEM

NODE PROBLEMS

ADDITIONAL DATA

User Experience: ERROR

Execution Time: 30 ms

CPU Time: 0 ms 0 %

Transaction Timestamp: 10/19/12 12:12:17 AM (server) 10/19/12 12:12:17 AM (agent)

Summary: [Error] - com.appdynamicspilot.webserviceclient.SoapUtils::There was an exception in checking out 5 : AxisFault: Exception occurred while trying to invoke service method createOrder http://localhost:8002/cart/services/OrderService?wsdl : AxisFault: Exception occurred while trying to invoke service method createOrder -

Error: Exception Message: Exception occurred while trying to invoke service method createOrder

Tier: ECommerce Server

Node: Node_8000

Business Transaction: Checkout

URL: /appdynamicspilot/ViewCart/sendItems.action

Session ID: C1EF6D085AA5CCB44A7BBA9878A43382

User Principal: No User Principal

Process ID: 8117

Thread Name: http-8000-Processor17

Thread ID: 42

Transaction Thresholds: Slow: 3.0x of standard deviation [1792.2704] for moving average [585.3495] for the last [120] minutes. Very Slow: 4.0x of standard deviation [1792.2704] for moving average [585.3495] for the last [120] minutes. Configure

Request GUID: 5ba5ea62-c629-4bf1-bc47-e2855118a1be

Export to PDF

(close)

You can use the Export to PDF button at the lower left to send this information to your colleagues.

Go back to the flow map and click the Inventory tier **Drill Down** button. You see the Call Drill Down of the Inventory tier error message.

The screenshot shows the 'Call Drill Down' window for a transaction. The title bar indicates 'Exe Time: 3 ms' and 'Timestamp: 10/19/12 12:12:17 AM'. The left sidebar contains navigation options: SUMMARY, SQL CALLS, HTTP PARAMS, COOKIES, USER DATA, ERROR DETAILS, HARDWARE / MEM, NODE PROBLEMS, and ADDITIONAL DATA. The main content area displays the following details:

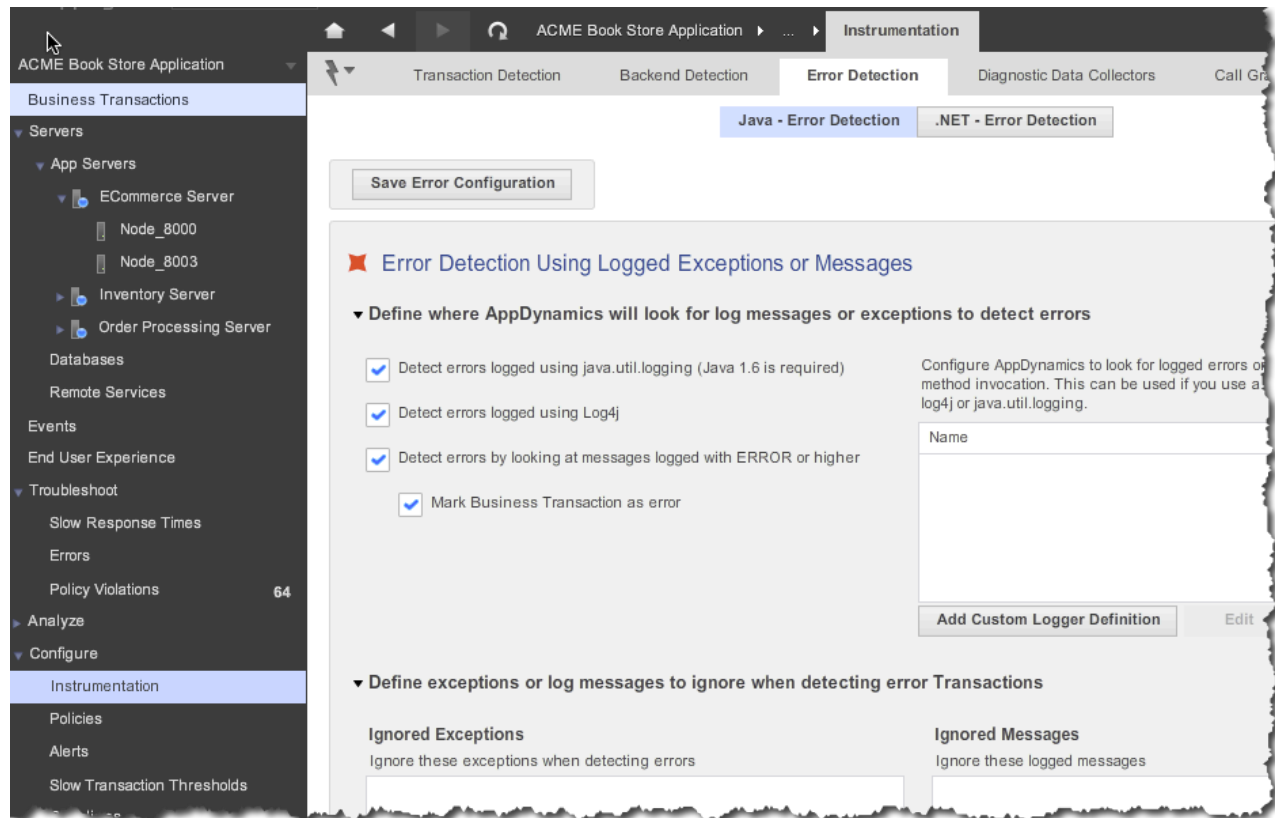
- User Experience:** ERROR (indicated by a red icon)
- Execution Time:** 3 ms
- CPU Time:** 0 ms 0 %
- Transaction Timestamp:** 10/19/12 12:12:17 AM (server) 10/19/12 12:12:17 AM (agent)
- Summary:** [Error] - org.apache.axis2.rpc.receivers.RPCMessageReceiver::Exception occurred while trying to invoke service method createOrder : InvocationTargetException com.appdynamics.inventory.OrderService : Error in creating order5 -
- Error:** Exception Message: null
- Tier:** Inventory Server
- Node:** Node_8002
- Business Transaction:** Checkout
- URL:** /cart/services/OrderService
- Session ID:** (not found)
- User Principal:** No User Principal
- Process ID:** 8153
- Thread Name:** http-8002-Processor18
- Thread ID:** 46
- Transaction Thresholds:** Slow: 3.0x of standard deviation [1392.9971] for moving average [296.60364] for the last [120] minutes. Very Slow: 4.0x of standard deviation [1392.9971] for moving average [296.60364] for the last [120] minutes. [Configure](#)
- Request GUID:** 5ba5ea62-c629-4bf1-bc47-e2855118a1be

At the bottom left, there is an 'Export to PDF' button and a '(close)' link.

Compare the two error messages.

See How Exceptions are Configured

AppDynamics provides application-level default configurations for detecting exceptions. In the left navigation pane click **Configure -> Instrumentation -> Error Detection**.

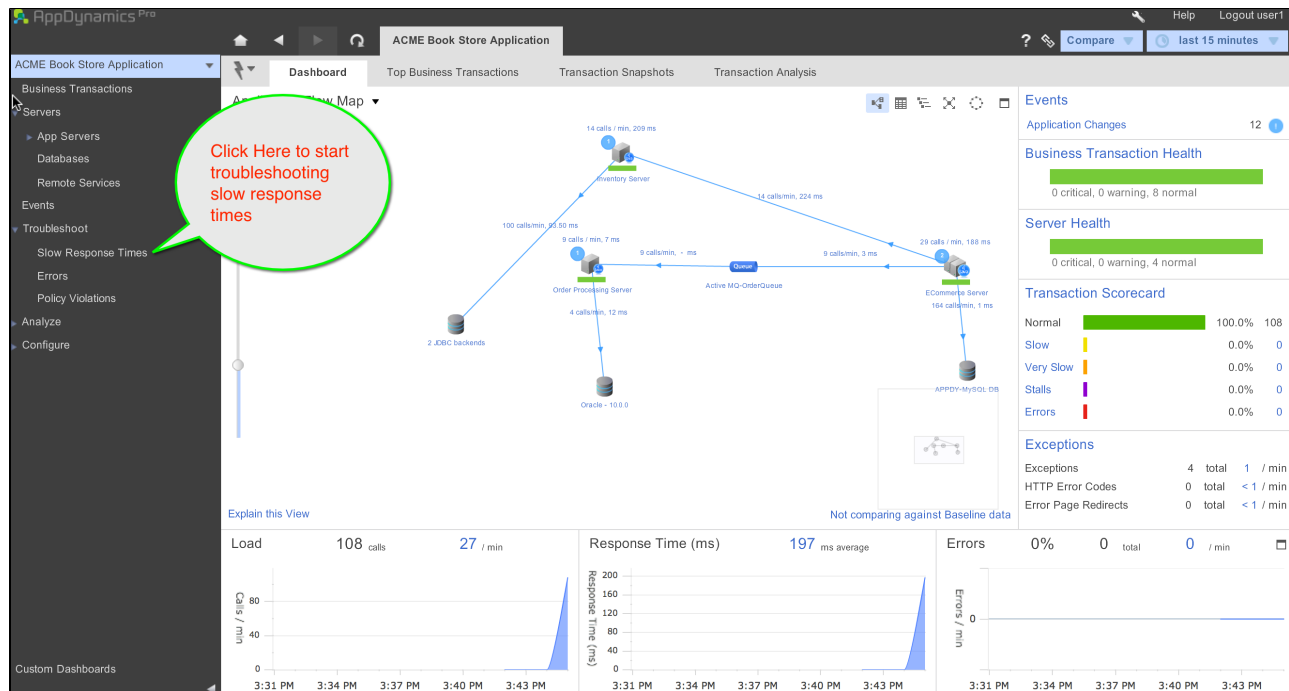


Learn More

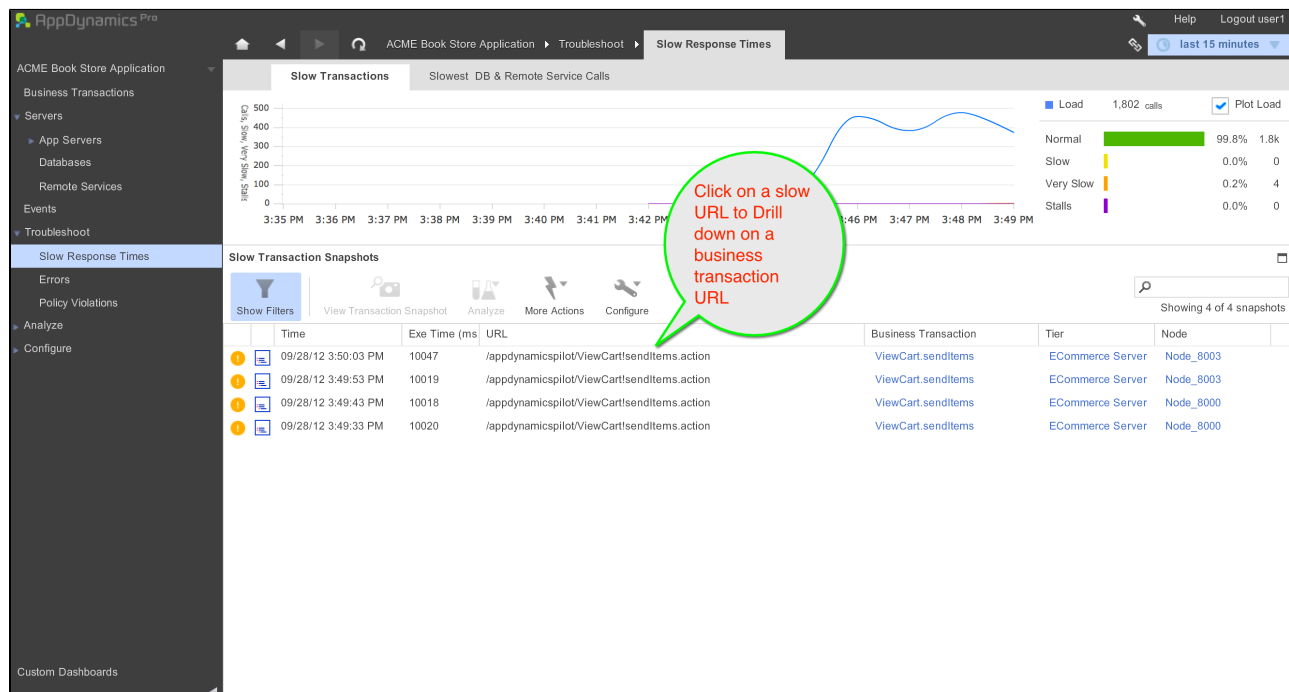
- [Troubleshoot Errors](#)
- [Configure Error Detection](#)

Tutorial for Java - Slow Transactions

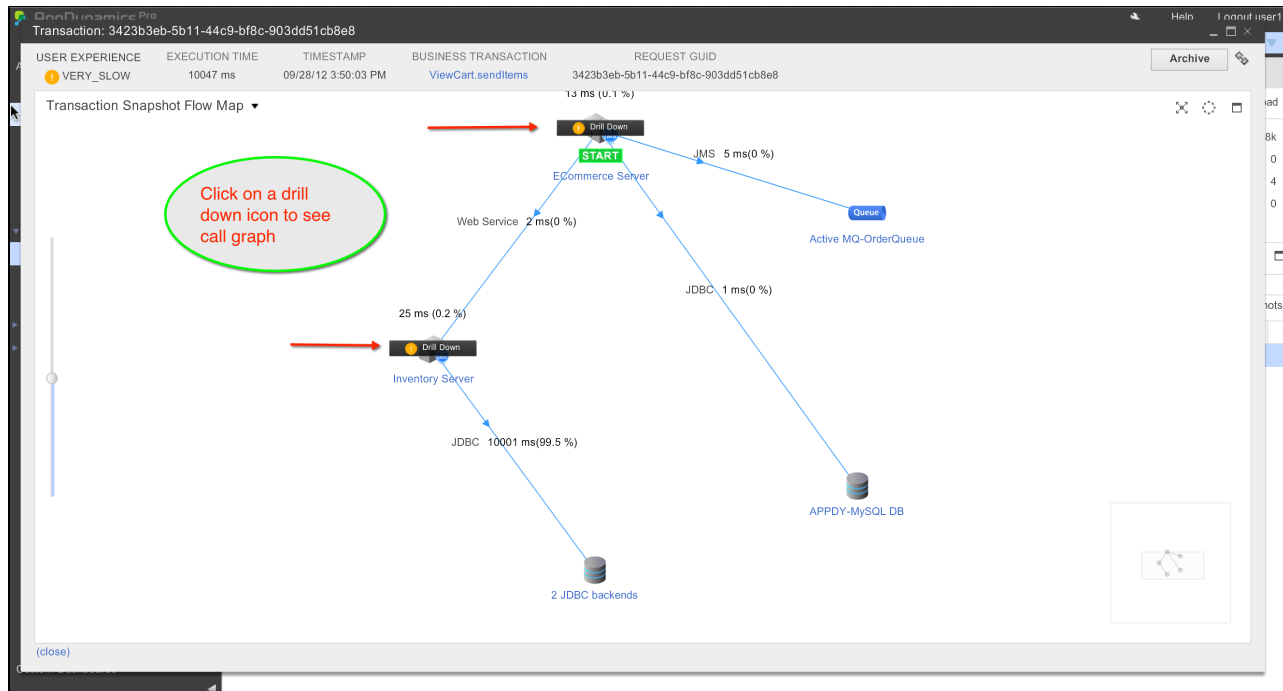
To begin troubleshooting, click **Troubleshoot -> Slow Response Times**:



To troubleshoot slow business transaction URLs, select the Slow Transactions tab and use the Transaction Snapshots pane:



Once you select the URL you will see a visualization of the transaction. You can drill into a call graph by clicking the drill-down icon.



Once you are in the call graph you can look for methods that have a significant response time. For example, the executeQuery method is responsible for 99% of response time:

Partial Call Graph

Execution Time: 10026 ms. Node Node_8002. Timestamp: 09/28/12 3:50:03 PM.

Call Drill Down. Exe Time: 10026 ms. Timestamp: 09/28/12 3:50:03 PM. BT: ViewCart.sendItems GUID: 3423b3eb-5b11-44c9-bf8c-903dd51cb8e8

Set as Root. Reset Root (?)

Show Filters

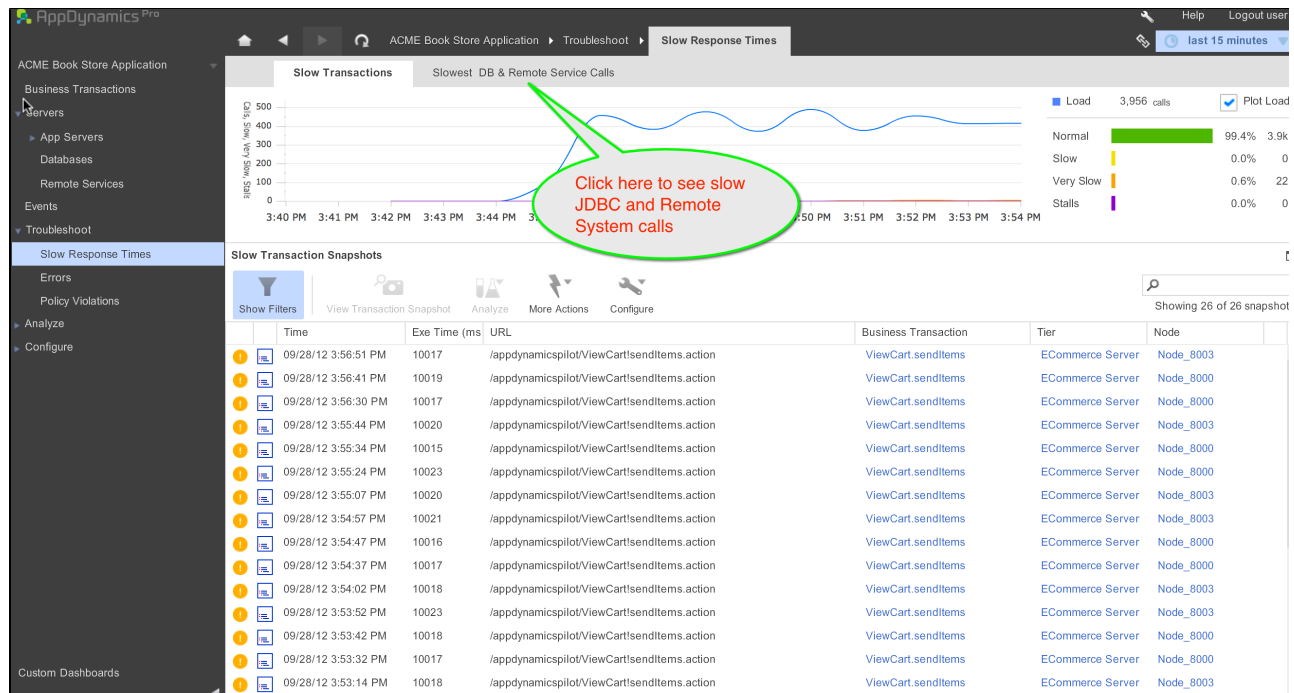
Name	Time (ms)	Exit Calls / Threads
java.lang.Thread.run:680	0 ms (self) 0 %	
HTTPServlet.service:729	0 ms (self) 0 %	
HTTPServlet.service:647	0 ms (self) 0 %	
Axis2 Webservice Servlet.doPost:116	0 ms (self) 0 %	
Web Service - org.apache.axis2.receivers.AbstractInOutSyncMessageReceiver.receive:39	0 ms (self) 0 %	
Web Service - org.apache.axis2.rpc.receivers.RPCMessageReceiver.invokeBusinessLogic:116	0 ms (self) 0 %	
Spring Bean - orderService.createOrder:16	0 ms (self) 0 %	
Proxy For Spring Bean - orderServiceTarget.createOrder	0 ms (self) 0 %	
Proxy For Spring Bean - orderServiceTarget.invoke	0 ms (self) 0 %	
Spring Bean - orderServiceTarget.createOrder:22	0 ms (self) 0 %	
Spring Bean - orderDao.createOrder:33	18 ms (self) 0.2 %	
com.appdynamics.inventory.QueryExecutor.executeSimplePS:61	0 ms (self) 0 %	
com.appdynamics.jdbc.MPreparedStatement.executeQuery:41	10005 ms (total) 99.8 %	JDBC

We find a very slow JDBC query

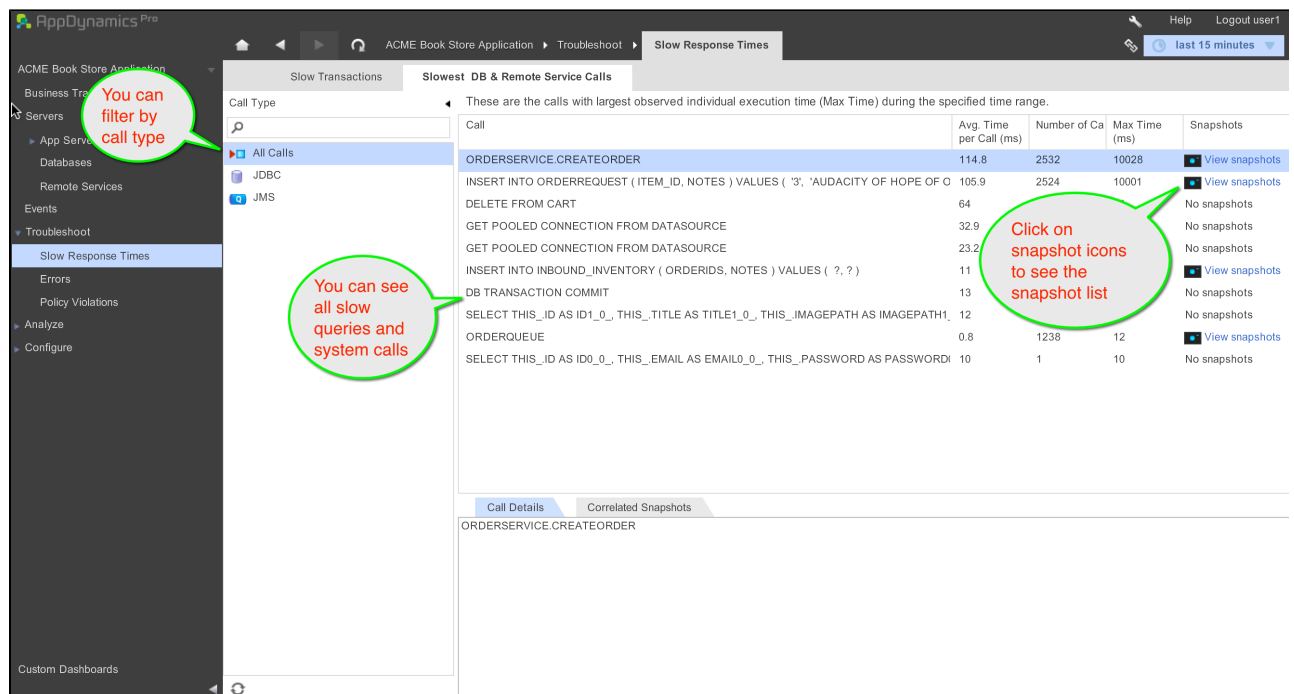
Export to PDF

Some packages have been excluded from this Call Graph

From the **Troubleshoot -> Slow Response Times** page, you can also select the **Slowest DB & Remote Service Calls** tab:



You can drill into the transaction snapshots from this tab to see the snapshot view:



For more information on resolving issues related to slow transactions, see [Troubleshoot Slow Response Times for Java](#).

Tutorial for Java - Troubleshooting using Events

- Troubleshooting with Events
 - How to Set up the Events List
 - How to Know Something is Not Quite Right

- How to Investigate
 - Investigating Errors
 - Investigating Stalled Business Transactions
 - Investigating Slow Business Transactions
 - Investigating Application Server Exceptions
 - Investigating Code Deadlocks
 - Investigating Application Change Events

Troubleshooting with Events

How to Set up the Events List

1. From the left navigation pane, click an application and then click **Events**.

✓ You can also access the **Events** window by clicking **Events** on the right side of the application dashboard.

2. In the **Events** window, use the filter criteria to pick which events you want to monitor. Click **Search**.
3. Set the time range.
4. Look for issues and anomalies.

How to Know Something is Not Quite Right

You see:

- Red (critical, policy violation)
- Purple (warning, stall)
- Orange (warning, very slow)
- Yellow (warning, slow)

How to Investigate

You drill down to the root cause of the problem in different ways depending on the type of event.

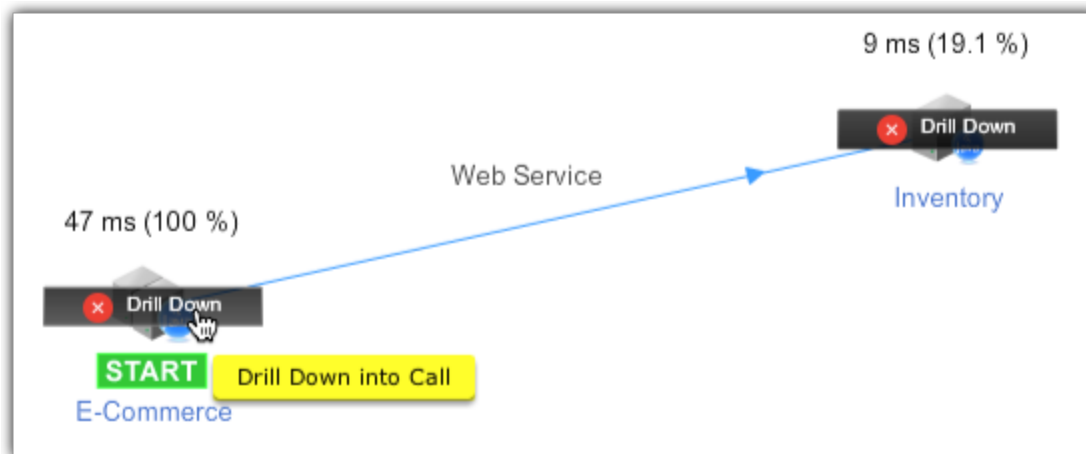
Investigating Errors

You can troubleshoot application issues by drilling down into errors.

1. In the **Events** window click an **Error**.



2. In the **Transaction Flow Map** click the Drill Down icon. If there are multiple drill down icons, select the one with the transaction that takes the most time.



3. In the **Call Drill Down** window click the **Summary** tab.

Call Drill Down. Exe Time: 47 ms Timestamp: 01/07/13 1:58:32 PM BT: Checkout GUID: 0cdcf690-e6a1-4c4d-8d2f-e53b693d0556

SUMMARY	User Experience	Execution Time	CPU Time	Transaction Timestamp	Summary	Error	Tier	Node	Business Transaction	URL	Session ID	User Principal	Process ID	Thread Name	Thread ID	Transaction Thresholds	Request GUID
SQL CALLS	ERROR	47 ms	0 ms 0 %	01/07/13 1:58:32 PM (server) 01/07/13 1:58:32 PM (agent)	[Error] - com.appdynamicspilot.webserviceclient.SoapUtils::There was an exception while trying to invoke service method createOrder http://localhost:8002/cart/services/OrderService?wsdl	Exception Message: Exception occurred while trying to invoke service method createOrder	E-Commerce	E-Commerce-Node-8000	Checkout	/appdynamicspilot/ViewCart/sendItems.action	6F9E4F18355CB2B4958E27CBF5A87DE0	No User Principal	7366	http-8000-Processor19	46	Slow: 350 ms. Very Slow: 700 ms.	0cdcf690-e6a1-4c4d-8d2f-e53b693d0556
HTTP PARAMS																	
COOKIES																	
USER DATA																	
ERROR DETAILS																	
HARDWARE / MEM																	
NODE PROBLEMS																	
ADDITIONAL DATA																	

4. Use the **Summary** information to troubleshoot issues. This information can also be exported to PDF.

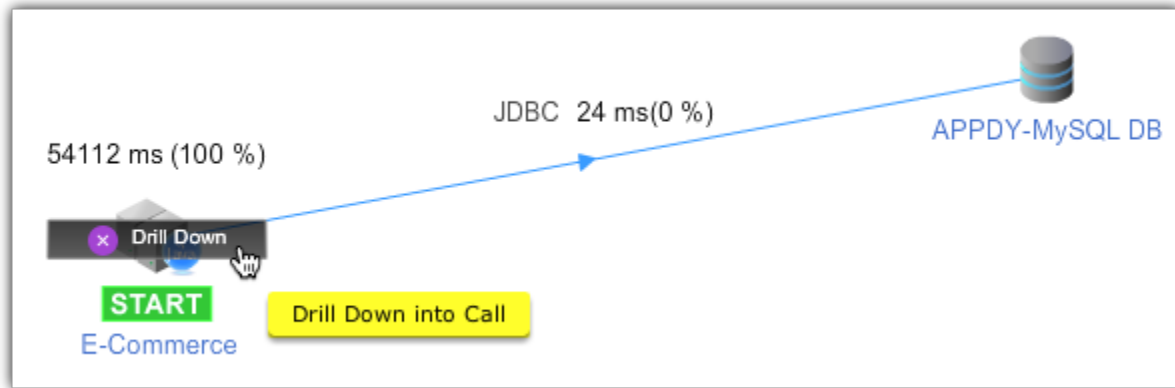
Investigating Stalled Business Transactions

You can troubleshoot business transactions by drilling down into stalled business transactions.

1. In the **Events** window click a **Slow Requests - Stalled** row.



2. In the **Transaction Flow Map** click the Drill Down icon.



3. In the **Call Drill Down** window click the **Summary** tab.

User Experience ✖ STALL

Execution Time 54136 ms

CPU Time 0 ms 0 %

Transaction Timestamp 01/07/13 1:57:41 PM (server) 01/07/13 1:57:41 PM (agent) ?

Summary **Request took higher than the stall threshold of [45000] ms -**

Tier E-Commerce

Node E-Commerce-Node-8000

Business Transaction Add to cart

Stack Dump

Thread Name:http-8000-Processor24
ID:51
Time:Mon Jan 07 21:58:26 UTC 2013
State:TIMED_WAITING
Priority:5

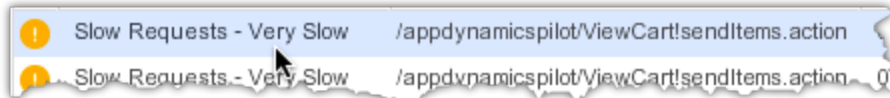
```
java.lang.Thread.sleep(Native Method)
com.appdynamics.pilot.action.CartAction.addToCart(CartAction.java:109)
sun.reflect.GeneratedMethodAccessor163.invoke(Unknown Source)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
com.opensymphony.xwork2.DefaultActionInvocation.invokeAction(DefaultActionInvocation.java:40)
com.opensymphony.xwork2.DefaultActionInvocation.invokeActionOnly(DefaultActionInvocation.java:229)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.java:229)
com.opensymphony.xwork2.interceptor.DefaultWorkflowInterceptor.doIntercept(DefaultWorkflowInterceptor.java:120)
com.opensymphony.xwork2.interceptor.MethodFilterInterceptor.intercept(MethodFilterInterceptor.java:48)
```

4. Use the **Summary** information to troubleshoot business transaction issues. This information can also be exported to PDF.

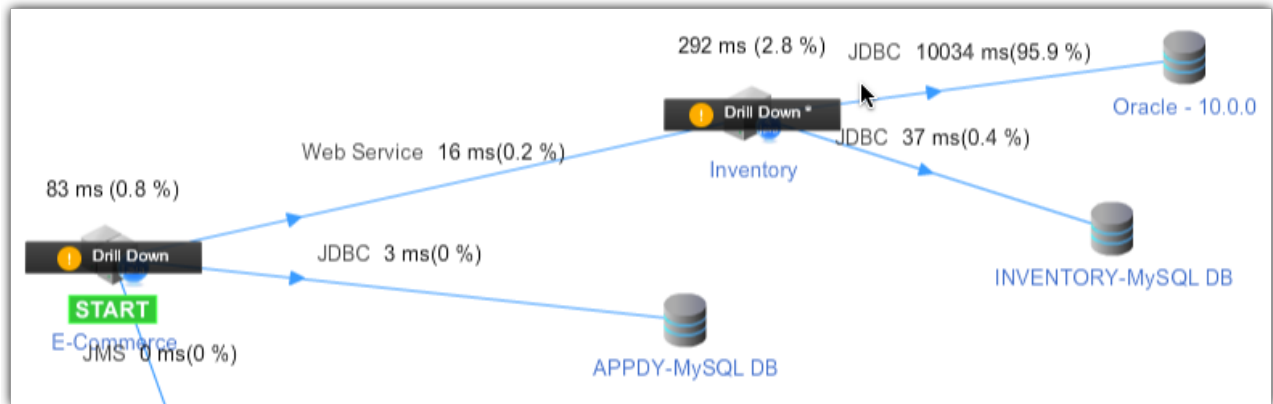
Investigating Slow Business Transactions

You can troubleshoot business transactions by drilling down into slow or very slow business transactions.

1. In the **Events** window click a **Slow Requests - Very Slow** or **Slow** row.



2. In the **Transaction Flow Map** click the **Drill Down** icon. If there are multiple drill down icons, select the one with the transaction that takes the most time.



If there is more than one call from the originating Tier, you will see the **Select a Call** window. In this case, proceed to step 3. Otherwise, skip to step 4.

3. In the **Select a Call** window click the slowest call.

Select a Call to Drill Down into

Multiple calls were made to this Tier as part of this Transaction.

Drill Down into Call Show: All Calls

	Exe Time (ms)	Summary	Exit Calls
!	10032 ms	[Web Service] call from E-Commerce	7 JDBC calls (10003 ms. max, 1429.0 ms. avg.)
✓	299 ms	[Web Service] call from E-Commerce	7 JDBC calls (17 ms. max, 2.4 ms. avg.)
✓	32 ms	[Web Service] call from E-Commerce	7 JDBC calls (14 ms. max, 2.0 ms. avg.)

4. In the **Call Drill Down** window click the **Hot Spots** tab to see the slowest methods.

This screen displays all of the method calls in the call graph sorted by time

Name	Method Time (ms)	External Calls
com.appdynamics.jdbc.MPreparedStatement.executeQuery:45	10005 ms (self) 99.7 %	JDBC
Spring Bean - transactionManager.doCommit:578	23 ms (self) 0.2 %	JDBC

Invocation Trace

```

AxisServlet.doPost:unknown (3ms self time, 10031 ms total time)
AbstractInOutSyncMessageReceiver.receive:39 (0ms self time, 10028 ms total time)
RPCMessageReceiver.invokeBusinessLogic:116 (0ms self time, 10028 ms total time)
OrderWebservices.createOrder:16 (0ms self time, 10028 ms total time)
OrderService$$EnhancerByCGLIB$$1ee8c32e.createOrder:unknown (0ms self time, 10028 ms total time)
OrderService$$FastClassByCGLIB$$e49d675f.invoke:unknown (0ms self time, 10005 ms total time)
OrderService.createOrder:22 (0ms self time, 10005 ms total time)
OrderDaoImpl.createOrder:33 (0ms self time, 10005 ms total time)
QueryExecutor.executeSimplePS:61 (0ms self time, 10005 ms total time)
MPreparedStatement.executeQuery:45 (10005ms self time, 10005 ms total time)
    
```

5. In this example, since the slow call is a database call you know you can click the **SQL Calls** tab to see the slowest SQL statement.

Query Type	Query	Avg. Time	Count
Insert	insert into OrderRequest (item_id, notes) values (?, ?)	10003	1
COMMIT	DB Transaction Commit	22	1

6. Use this information to diagnose transaction issues. This information can also be exported to PDF.

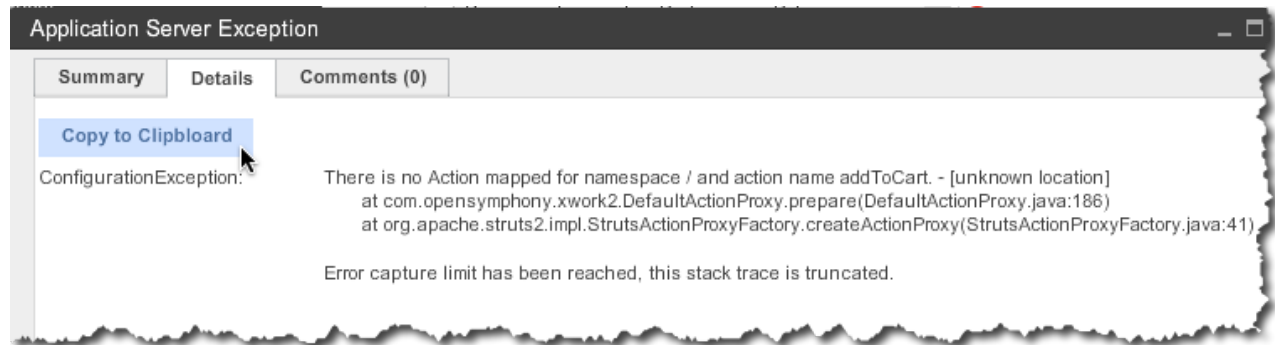
Investigating Application Server Exceptions

You can troubleshoot application server issues by drilling down into application server exceptions.

1. In the **Events** window click an **Application Server Exception**.

Application Server Exception	org.apache.struts2.dispatcher.Dispatch.
Slow Requests - Stalled	/appdynamicspilot/ViewCart!addToCart.a

2. In the **Application Server Exception** window, click the **Details** tab.



3. Use this information to troubleshoot application server issues. Use the **Copy to Clipboard** button to save the exception details.

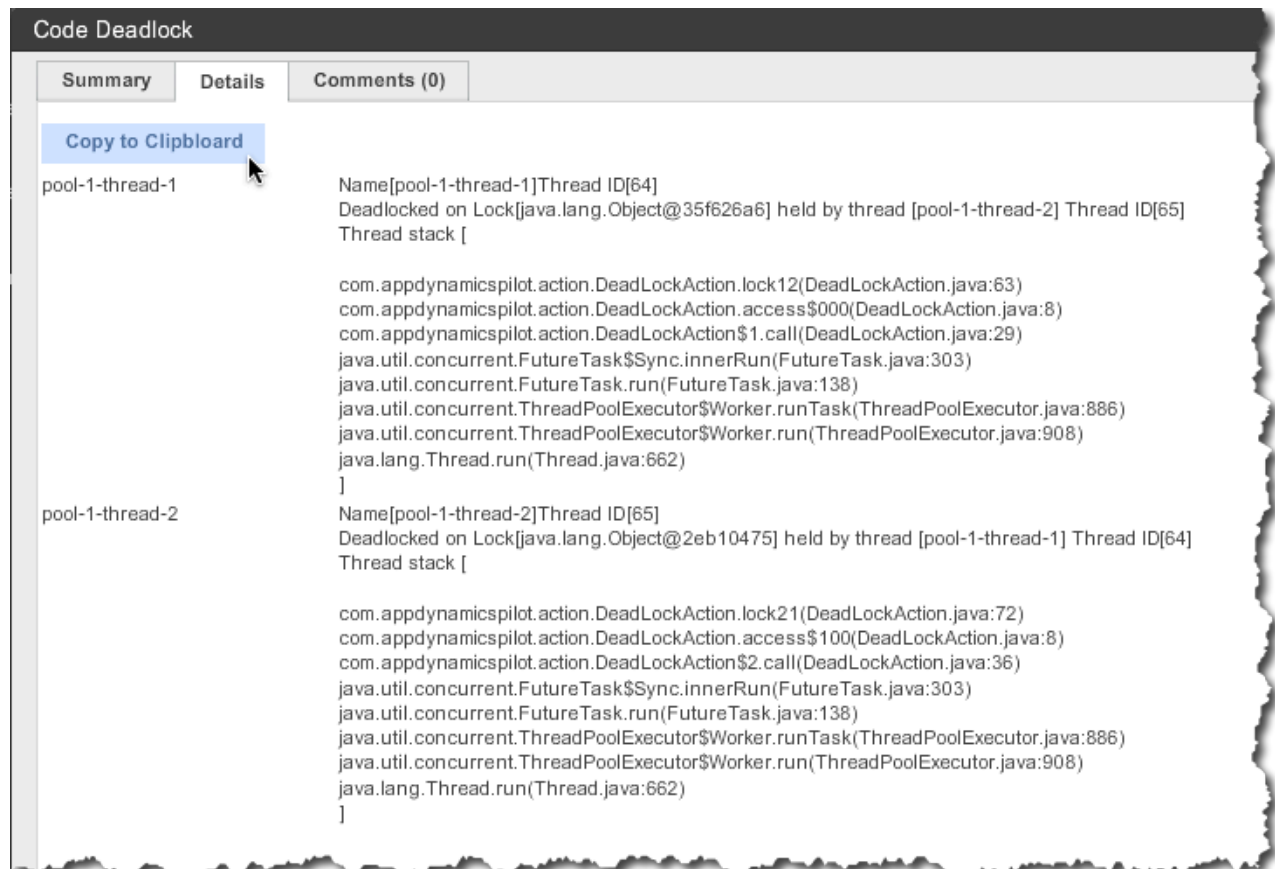
Investigating Code Deadlocks

You can troubleshoot code deadlocks by drilling down into a code deadlocks.

1. In the **Events** window click a **Code Deadlock**.



2. In the **Code Deadlock** window click the **Details** tab.



3. Use this information to troubleshoot code deadlock issues. Use the **Copy to Clipboard** button to save the deadlock details.

Investigating Application Change Events

You can view application changes by drilling down into application change events.

- ✓ By default AppDynamics reports events when applications are deployed, app servers are restarted, and configuration parameters are changed. Since these are not problems, they are indicated by a blue icon.

1. Click a change event to see a summary and details, for example:

1	App Server Restart	Application Server JVM was re-started Node: Inve...	01
1	Application Configuration Change	Application Server environment variables changed	01
1	Application Configuration Change	Application Server VM system properties changed	01

2. Use this information to view application changes. Use the **Copy to Clipboard** button to save the change details.



Troubleshoot .NET Application Problems

Troubleshoot Slow Response Times for .NET

- How Do You Know Response Time is Slow?
 - You Received an Alert
 - You are Viewing the Application Dashboard for a Business Application
- Initial Troubleshooting Steps
- Troubleshooting Methodology
 - Step 1 - All nodes?
 - Step 2 - Most business transactions?
 - Step 3 - Backend problem?
 - Step 4 - CPU saturated?
 - Step 5 - Significant garbage collection activity?
 - Step 6 - Memory leak?
 - None of the above?

How Do You Know Response Time is Slow?

There are two primary ways you can learn that your application's response time is slow: receiving

an alert and looking at an Application Dashboard.

You Received an Alert

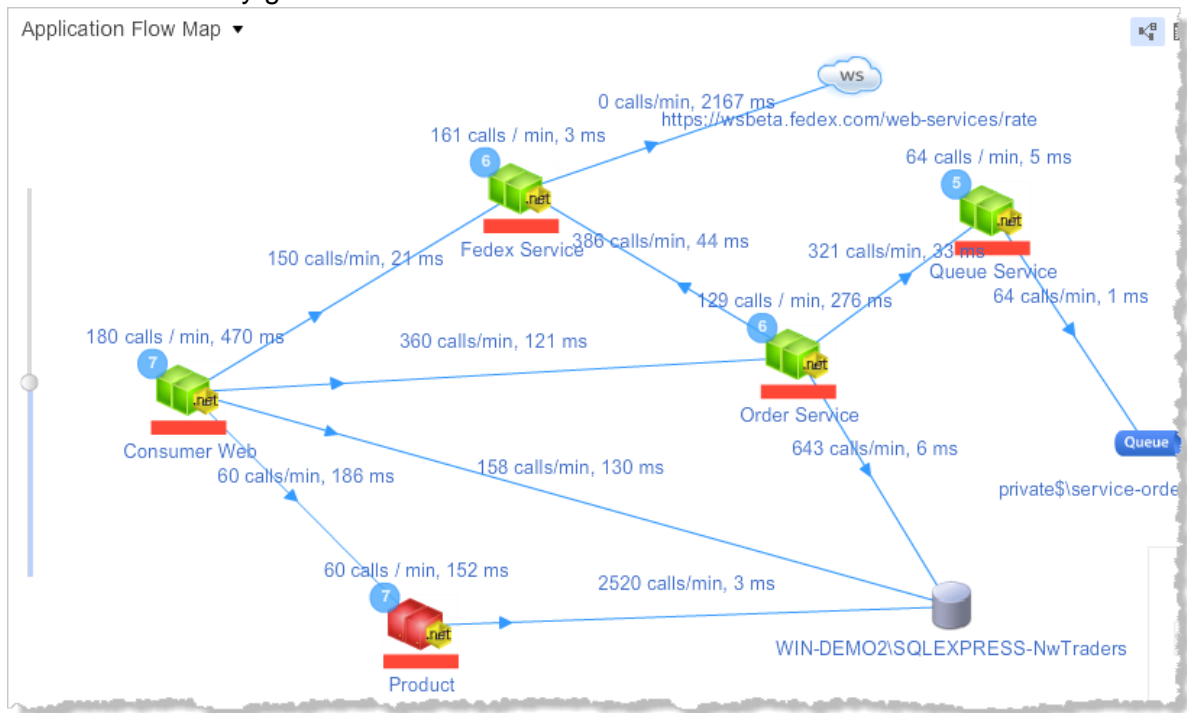
If you have received an email alert from AppDynamics that was configured through the use of [Health Rules](#), [Policies](#), [Actions](#) or [Workflow Overview](#), the email message provides a number of details about the problem that triggered the alert. See [Email Notifications](#).

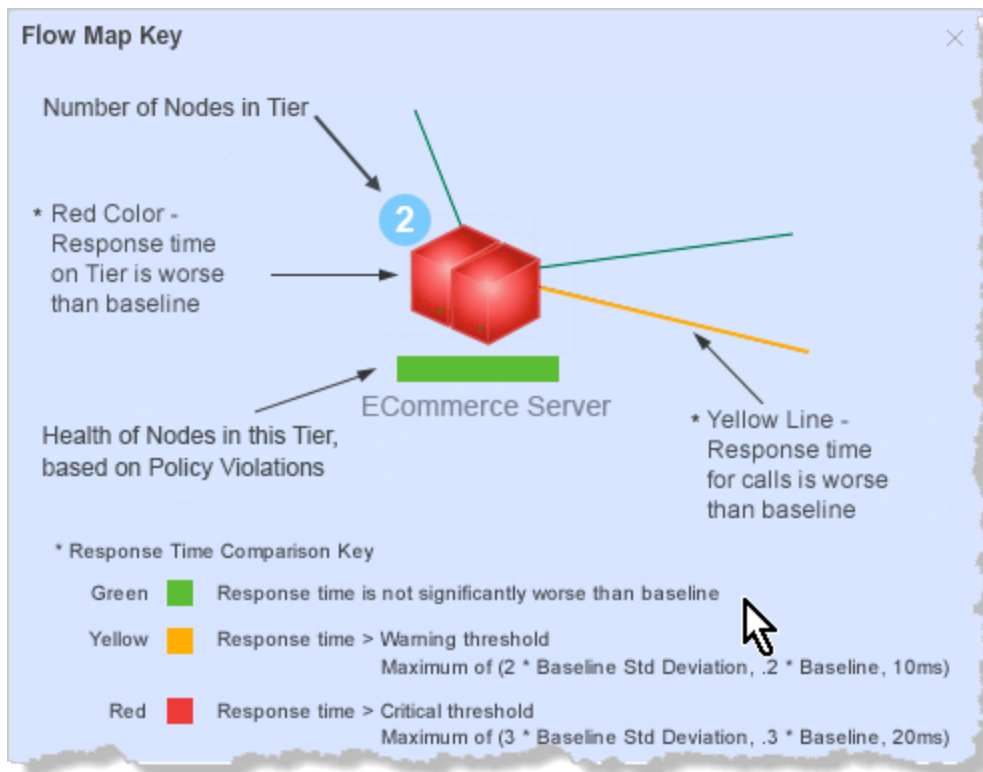
If the problem is related to slow response time, see [Initial Troubleshooting Steps](#).

You are Viewing the Application Dashboard for a Business Application

NOTE: If you know the slow response time relates to a particular business transaction, e.g. an internal tester reported "Searching for a hotel is slow," skip to the answer No in [Step 2](#).

1. Display the Application Dashboard (flow map). Look for lines that are yellow or red or tiers that are not entirely green.





- If multiple tiers are red or yellow, the problem might be related to a backend (database or other remote service). Skip to [Step 3](#).
- Otherwise, begin troubleshooting with [Initial Troubleshooting Steps](#).

Need more help?

- [Application Dashboard](#)
- [Flow Maps](#)

Initial Troubleshooting Steps

In some cases, the source of your problem might be easily diagnosed by choosing **Troubleshoot -> Slow Response Times** in the Navigation Pane. See [Troubleshoot Slow Response Times](#).

If you've tried to diagnose the problem using those techniques and haven't found the problem, use the following troubleshooting methodology to find other ways to determine the root cause of your issue.

Troubleshooting Methodology

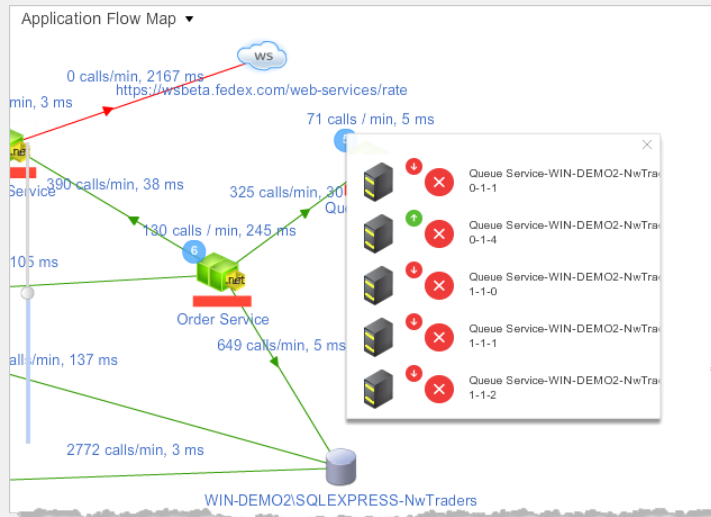
Step 1 - All nodes?

Is the problem affecting all nodes in a slow tier?

How do I know?

How do I know if the problem is affecting all nodes?

- In the Application or Tier Flow Map, click the number that represents how many nodes are in the tier. This provides a quick overview of the health of each node in the tier. The small circle icon indicates whether the server is up with the agent reporting, and the larger circle icon indicates Health Rule violation status.

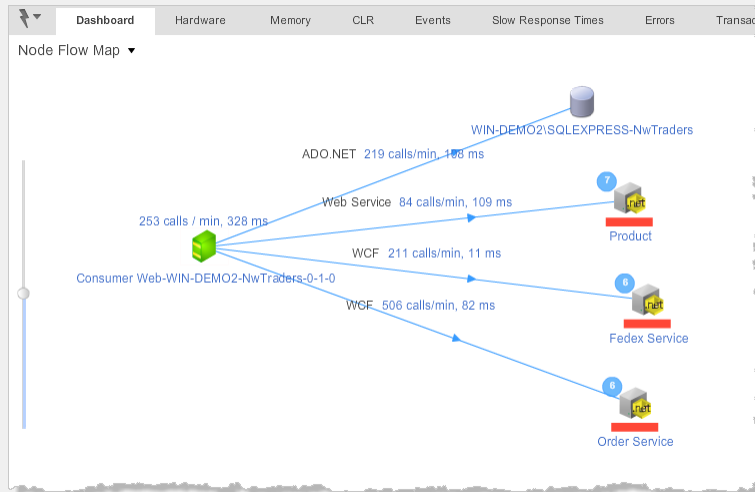


If all the nodes are yellow or red, the answer to the question in Step 1 is Yes. Otherwise, the answer is No.

Yes – Go to [Step 2](#).

No – The problem is either in the node's hardware or in the way the software is configured on the node. If only one node in a tier is affected, the problem is probably not related to the application code.

- In the left navigation pane, click **Servers -> App Servers -> <slow tier> -> <problematic node>** to display the Node Dashboard flow map.



- Click the Dashboard tab to get a view of the overall health of the node.
- Click the Hardware tab to see if there is a hardware resource issue.
- Click the Memory tab and sort on various column headings to determine if there is a shortage of memory or other memory issue.

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Node Dashboard](#)

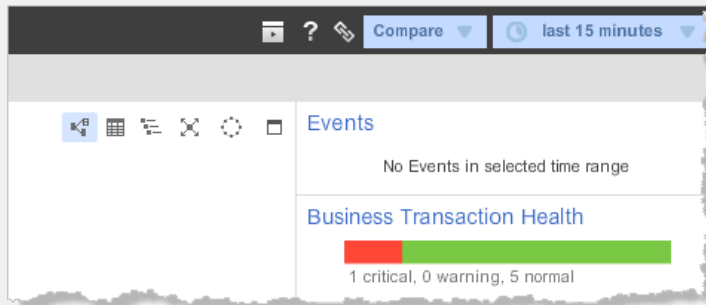
Step 2 - Most business transactions?

Is the problem affecting most of the business transactions?

How do I know?

How do I know if the problem is affecting most of the business transactions?

1. On the Application Dashboard, look at the Business Transaction Health pane on the right side of the screen.



If the bar representing business transaction health is primarily yellow or red, the answer to the question in Step 2 is Yes. Otherwise, the answer is No.

Yes – Go to [Step 3](#).

No –

1. In the left navigation pane, click **Business Transactions**.
2. Sort by Health, Server Time, or other column headings to find the business transaction that is experiencing issues.
3. Double-click the problematic business transaction to see its Dashboard, then use the tabs to diagnose the problem.



You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- Business Transaction Dashboard
- Business Transaction Monitoring
- Business Transactions List
- Transaction Snapshots

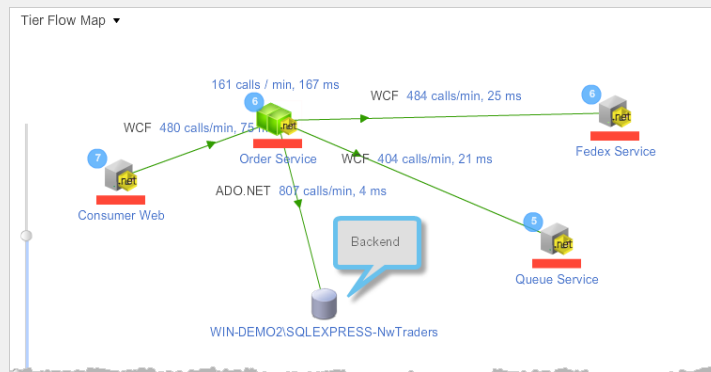
Step 3 - Backend problem?

Are the nodes linked to a backend (database or other remote service) that might be causing your problem?

▼ How do I know?

How do I know if the nodes are linked to a backend (database or other remote service) that might be causing my problem?

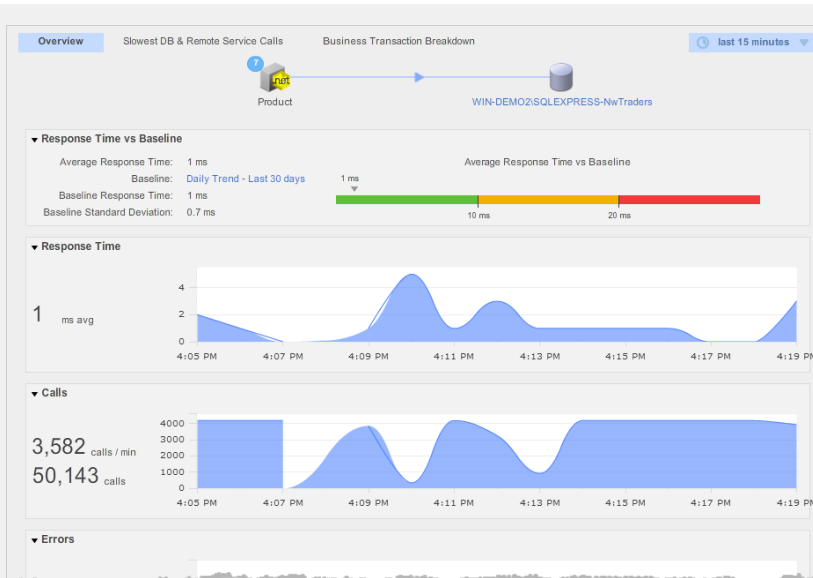
- Display the Tier Flow Map. If any nodes are linked to a backend, links to those backends are displayed in the flow map.



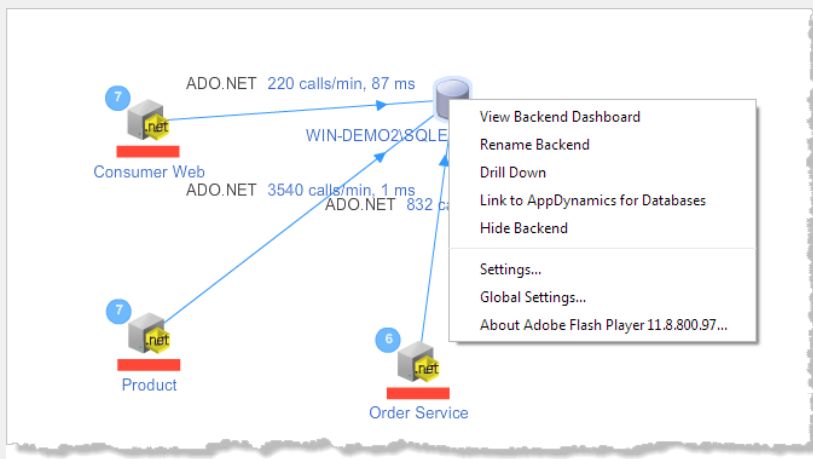
If a backend or the line connecting to a backend is red, the answer to the question in Step 3 is Yes. Otherwise, the answer is No.

Yes –

- Click the line connecting to the backend to see an information window about the backend. The contents of the information window vary depending on the type of backend. Use the various tabs to find the source of the issue.



- If the backend is a database, right-click the database icon. You have a number of options that let you see the dashboard, drill down, etc. If you have AppDynamics for Databases, choose Link to AppDynamics for Databases. You can use AppDynamics for Databases to diagnose database issues.



You have isolated the problem and don't need to continue with the rest of the steps below.

No – Go to [Step 4](#).

Need more help?

- [Backend Monitoring](#)
- [Configure Backend Detection for .NET](#)
- [AppDynamics for Databases](#)

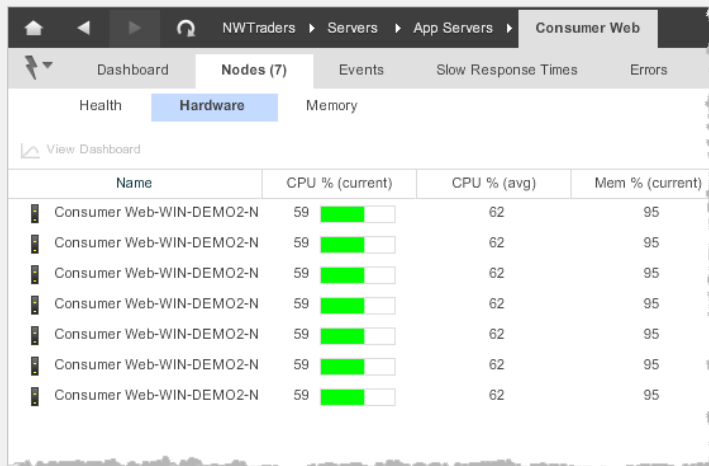
Step 4 - CPU saturated?

Is the CPU of the CLR saturated?

▼ [How do I know?](#)

How do I know if the CPU of the CLR is saturated?

1. Display the Tier Flow Map.
2. Click the Nodes tab, and then click the Hardware tab.
3. Sort by CPU % (current).



Name	CPU % (current)	CPU % (avg)	Mem % (current)
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95

If the CPU % is 90 or higher, the answer to the question in Step 4 is Yes. Otherwise, the answer is No.

Yes – Go to [Step 5](#).

No – Review various metrics in the Metric Browser to pinpoint the problem.

In the left navigation pane, click **Servers -> App Servers -> <slow tier>**. Review these metrics in particular:

- ASP.NET -> Application Restarts
- ASP.NET -> Request Wait Time
- ASP.NET -> Requests Queued
- CLR -> Locks and Threads -> Current Logical Threads
- CLR -> Locks and Threads -> Current Physical Threads
- IIS -> Number of working processes
- IIS -> Application pools -> <Business application name> -> CPU%
- IIS -> Application pools -> <Business application name> -> Number of working processes
- IIS -> Application pools -> <Business application name> -> Working Set

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Monitor .NET Applications](#)

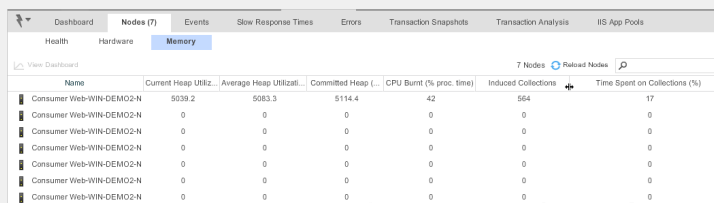
Step 5 - Significant garbage collection activity?

Is there significant garbage collection activity?

▼ [How do I know?](#)

How do I know if there is significant garbage collection activity?

1. Display the Tier Flow Map.
2. Click the Nodes tab, and then click the Memory tab.
3. Sort by Time Spent on Collections (%) to see what percentage of processing time is being taken up with garbage collection activity.



Name	Current Heap Utiliz.	Average Heap Utiliz.	Committed Heap	CPU Burnt (% proc. time)	Induced Collections	Time Spent on Collections (%)
Consumer Web-WIN-DEMO2-N	\$039.2	\$083.3	\$114.4	42	564	17
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0

If Time Spent on Collections (%) is higher than acceptable (say, over 40%), the answer to the question in Step 5 is Yes. Otherwise, the answer is No.

Yes – Go to [Step 6](#).

No – Use your standard tools to produce memory dumps; review these to locate the source of the problem.

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Monitor .NET Applications](#)

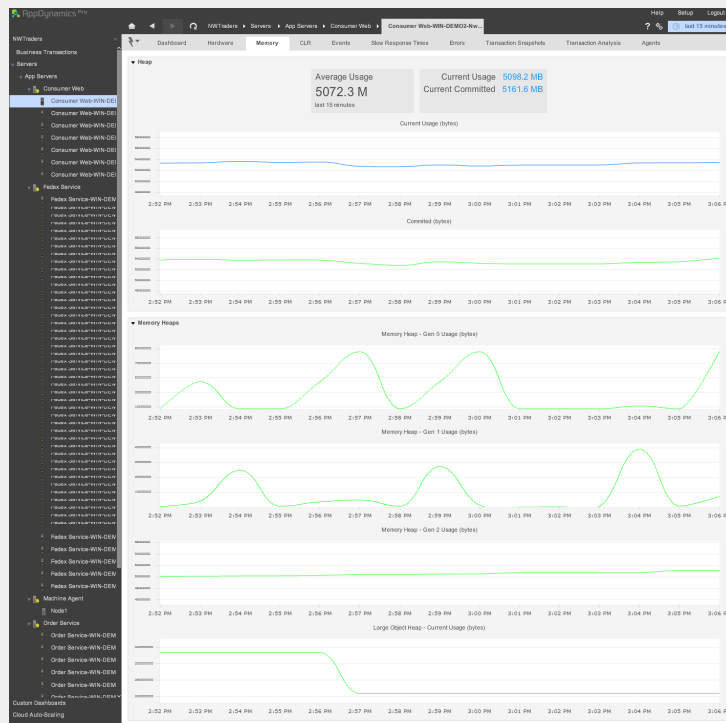
Step 6 - Memory leak?

Is there a memory leak?

✓ How do I know?

How do I know if there is a memory leak?

1. From the list of nodes displayed in the previous step (when you were checking for garbage collecting activity), double-click a node that is experiencing significant GC activity.
2. Click the Memory tab, then review the committed bytes counter and the size of the Gen0, Gen1, Gen2 and large heaps.



If memory is not being released (one or more of the above indicators is trending upward), the answer to the question in Step 6 is Yes. Otherwise, the answer is No.

Yes – Use your standard tools for troubleshooting memory problems. You can also review ASP.NET metrics; click **Servers** -> **App Servers** -> **<slow tier>** -> **ASP.NET**.

No – Use your standard tools to produce memory dumps; review these to locate the source of the problem.

Whether you answered Yes or No, you have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- [Monitor .NET Applications](#)

None of the above?

If slow response time persists even after you've completed the steps outlined above, you may need to perform deeper diagnostics.

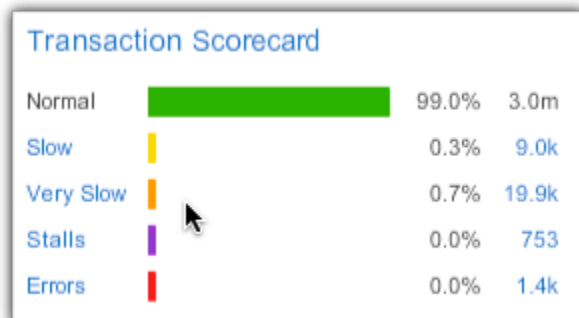
If you can't find the information you need on how to do so in the AppDynamics documentation, consider posting a note about your problem in a community discussion topic. These discussions are monitored by customers, partners, and AppDynamics staff. Of course, you can also contact AppDynamics support.

Need more help?

- [AppDynamics Pro Documentation](#)
- [Community Discussion Boards](#) (If you don't see AppDynamics Pro as a topic, click Sign In at the upper right corner of the screen.)

Troubleshoot PHP Application Problems

The [Dashboards](#) show you when problems occur and you need to take action. For example the [Transaction Scorecard](#) monitors business transactions and categorizes their performance according to [thresholds](#).



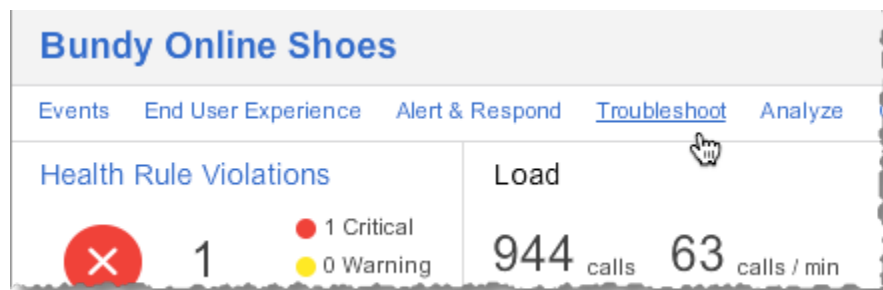
Troubleshooting Your Application

For common troubleshooting scenarios see:

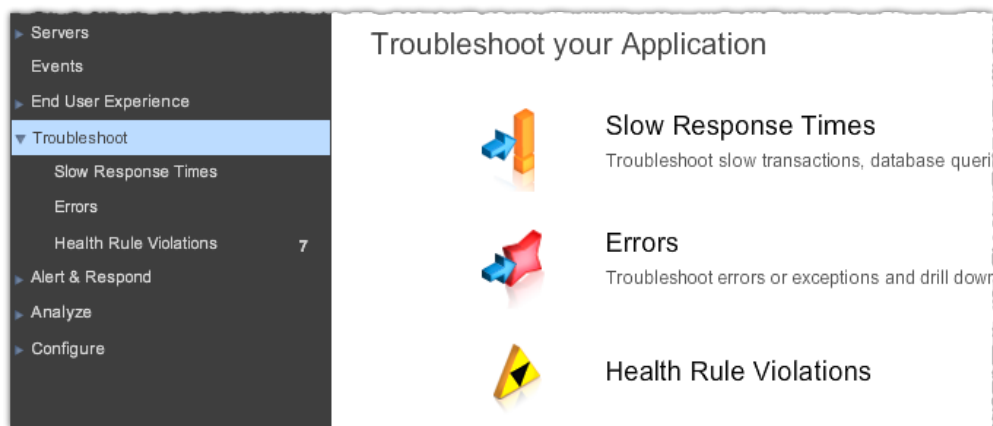
- [Troubleshoot Slow Response Times for PHP](#)
- [Troubleshoot Errors for PHP](#)
- [Troubleshoot Health Rule Violations](#)

When AppDynamics indicates a problem you can easily go right into troubleshooting mode.

- From the All Applications dashboard click the Troubleshoot link.



- Alternatively, after you have selected the application, from the left navigation menu click **Troubleshoot**.



For more information about troubleshooting see:

- [Troubleshoot Slow Response Times for PHP](#)
- [Troubleshoot Errors for PHP](#)
- [Troubleshoot Health Rule Violations](#)
- [Troubleshoot Node Problems](#)

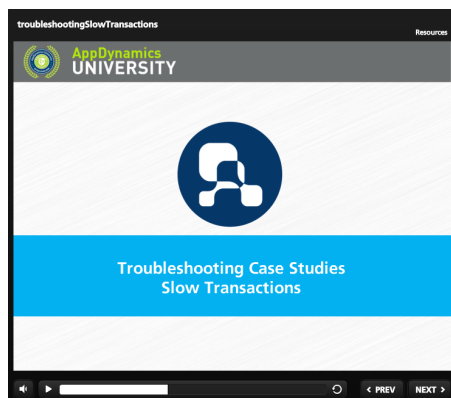
- [Transaction Snapshots](#)
- [Call Graphs](#)
- [Diagnostic Sessions](#)
- [Analyze](#)

Troubleshoot Slow Response Times for PHP

- [Slow and Stalled Transactions](#)
 - [To troubleshoot slow and stalled transactions](#)
- [Slow Database and Remote Service Calls](#)
 - [To troubleshoot slow database and remote service calls](#)
- [Learn More](#)

When you click **Troubleshoot -> Slow Response Times** the Slow Response Times window opens showing two tabs. You can drill down into transaction issues in the [Slow Transactions](#) tab and into database or remote services issues in the [Slowest DB & Remote Services](#) tab.

This two minute interactive video traces the typical steps of identifying the cause of a slow transaction.



Slow and Stalled Transactions

There are many reasons why a business transaction may be slow or stalled. The Slow Response Times tab helps you find the root cause whether that be resource contention, deadlock, race condition, or something else.

By default AppDynamics considers a slow transaction one that lasts longer than 3 times the standard deviation for the last two hours and a very slow transaction 4 times the baseline for the last two hours.

By default AppDynamics considers a transaction that lasts longer than 45 seconds (4500 milliseconds) to be stalled.

You can configure these thresholds to better match your environment. See [Thresholds](#) and [Configure Thresholds](#).

To troubleshoot slow and stalled transactions

1. Click **Troubleshoot -> Slow Response Times**.

You can also access this information from tabs in the various dashboards.

2. Click the **Slow Transactions** tab if it is not selected.

In the upper pane AppDynamics displays a graph of the slow, very slow, and stalled transactions for the time period specified in the Time Range drop-down menu. Click the Plot Load checkbox to see the load.

In the lower pane AppDynamics displays the transaction snapshots for slow, very slow, and stalled transactions.

Click the Exe Time column to sort the transactions from slowest to fastest.



To drill down, select a snapshot from the list and click **View Transaction Snapshot**. A transaction snapshot shows the details of an instance of a business transaction, including a call graph that helps you identify the root cause of the slow response time. See [Transaction Snapshots](#).

Slow Database and Remote Service Calls

Although AppDynamics does not instrument database and remote service servers directly, it collects metrics about calls to these backends from the instrumented app servers. This allows you to drill down to the root cause of slow database and remote service calls.

To troubleshoot slow database and remote service calls

1. Click **Troubleshoot -> Slow Response Times**.

You can also access this information from tabs in the various dashboards.

2. Click the **Slowest DB & Remote Service Calls** tab if it is not selected.

These are the calls with largest observed individual execution time (Max Time) during the specified time range.

Call	Avg. Time per Call	Number of Ca	Max Time (ms)	Snapshots
SELECT COUNT(1) COUNT FROM CUSTOMER C1, CUSTOMER C2	17879.2	74	110371	No snapshots
HTTP://WWW.SHOEWAREHOUSE.COM:8079/SHOEWAREHOUSE/SHOEWAR	4928.4	3565	5064	View snapshots
HTTP://EC2-54-214-253-138.US-WEST-2.COMPUTE.AMAZONAWS.COM:808	2600.6	1396	9819	No snapshots
HTTP://PAYMENT.VISA.COM/HEALTH/PAYMENT	1893.8	1215	12663	No snapshots
HTTP://API.FEDEX.COM/HEALTH/SHIPPING	1744.6	1221	15433	No snapshots
GET DB CONNECTION	325.4	5383	6614	No snapshots
GET DB CONNECTION	238.4	324	4910	No snapshots

3. In the Call Type panel select the type of call for which you want to see information or select All Calls.

The Call panel displays the call or query with the average time per call, number of calls, and maximum execution time (Max Time) for the calls with the longest execution time.

If transaction snapshots are available for a slow call, you can click **View Snapshots** link or you can select the call and click the **Correlated Snapshots** tab in the lower panel. From there you can select a snapshot and click **View Transaction Snapshot** to drill down to the root cause of the slow call.

View Transaction Snapshot

	Time	Exe Time (ms)	URL	Business Transaction	Tier	Node
	01/21/14 5:06:27 PM	5005	/Bur	/BundyBackend/search	Inventory	InventoryNode
	01/21/14 5:14:03 PM	5005	/Bur	/BundyBackend/search	Inventory	InventoryNode

Contains a partial Call Graph

See [Transaction Snapshots](#).

Learn More

- [Transaction Snapshots](#)
- [Configure Thresholds](#)

Troubleshoot Errors for PHP

- [Error Transactions and Exceptions](#)
 - [To Troubleshoot Error Transactions](#)
 - [To Troubleshoot Exceptions](#)
- [Learn More](#)

Identifying and troubleshooting errors in your application.

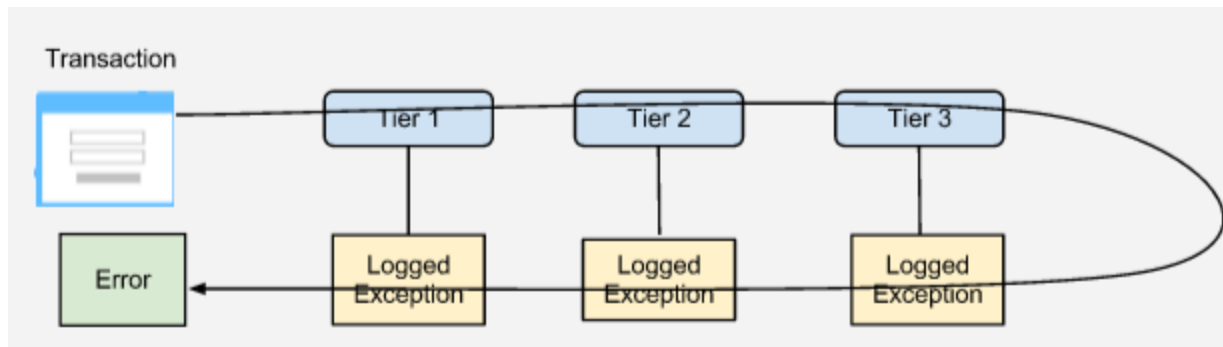
Error Transactions and Exceptions

An error transaction is a business transaction that experienced an error during the transaction execution. The error can be:

- A runtime error reported by the PHP server and captured by the agent. These include Fatal Errors, Warning and Notices thrown by the PHP runtime. These types of errors do not provide a stack trace.
- Certain exceptions thrown by the application. These include unhandled exceptions (when an exception is thrown and there is no **try** block) and handled exceptions during an exit call (when there is a **try** block during an exit call). These types of errors do provide a stack trace.

If a transaction experiences an error, it is counted as an error transaction and not as a slow, very slow or stalled transaction even if the transaction was also slow or stalled.

There is not a one-to-one correspondence between the number of errors and the number of exceptions. For example, a business transaction may experience a single code 500 error in which several exceptions were logged as the transaction passed through multiple tiers.



You can configure the types of errors that AppDynamics detects as well as the types of exceptions to ignore. See [Configure Error Detection for PHP](#).

To Troubleshoot Error Transactions

1. Click **Troubleshoot -> Errors** in the left navigation panel.

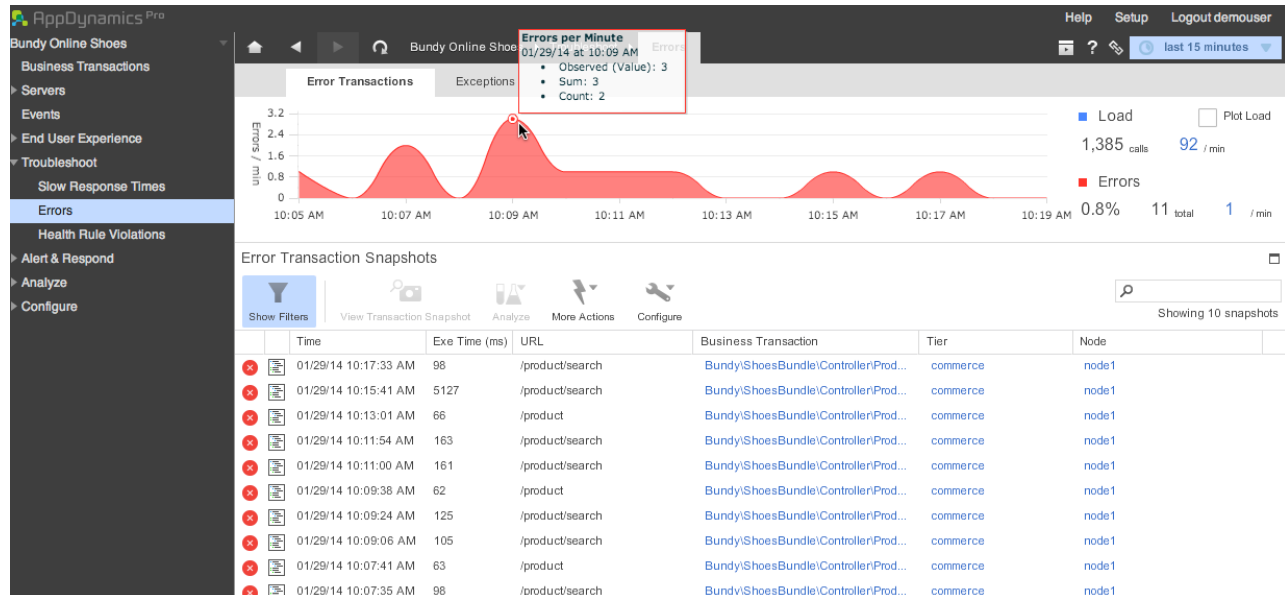
The error viewer opens.

2. Click the **Error Transactions** tab if it is not already selected.
3. From the time range drop-down menu select the time range for which you want to view information about error transactions.

A graph of the error transactions displays at the top of the viewer. You can get an exact count of the errors per minute at a particular point in time by hovering with your pointing device on the line in the graph.

To the right of the graph is a summary of the load and the error transactions.

Check the Plot check box if you want the graph at the top of the viewer to show the load over the selected time period. Clear this check box if you want the graph to show only the error transactions.

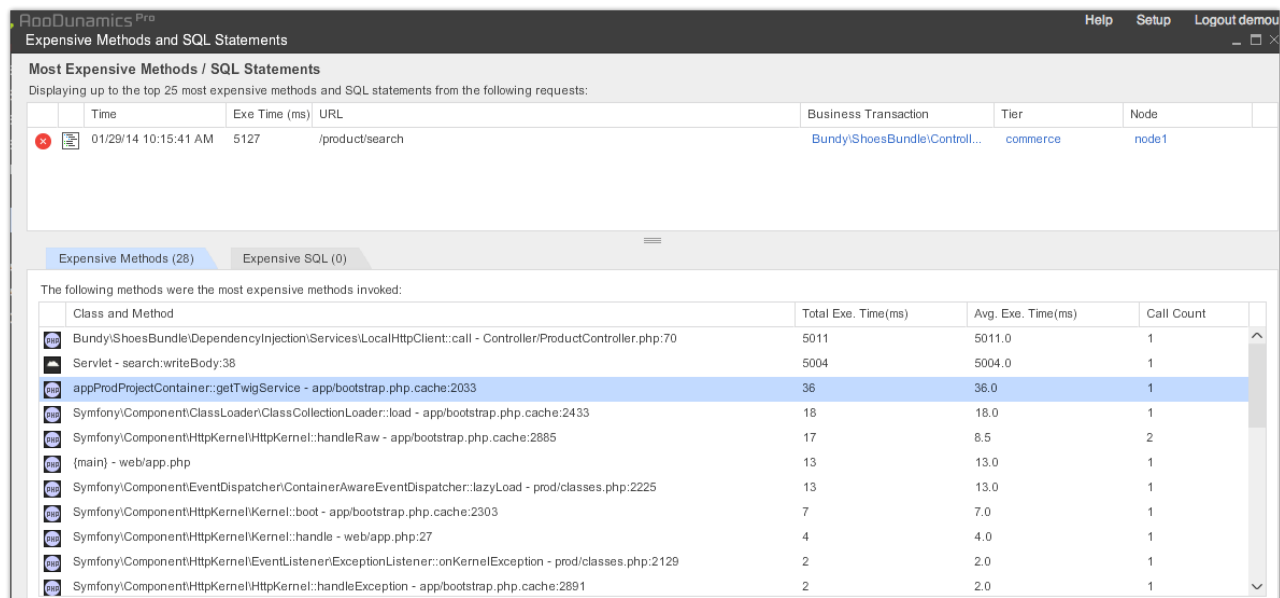


4. The error transaction snapshots are listed in the lower part of the viewer. To filter this list click **Show Filters** and select the filter criteria.

5. To examine the root cause of an error, select the snapshot from the list and click **View Transaction Snapshot**. See [Transaction Snapshots](#) for information about examining snapshots.

6. To identify the most expensive calls or queries, select a snapshot from the list and click **Analyze** and then click **Identify the most expensive calls / SQL statements in a group of snapshots**.

The Most Expensive Methods / SQL Statements viewer opens.



7. In the lower panel click the **Expensive Methods** tab to view the methods with their total and average execution times and call counts. Click the **Expensive SQL** tab to view the queries with their counts and execution times.

To Troubleshoot Exceptions

1. Click **Troubleshoot -> Errors** in the left navigation panel.
2. Click the **Exceptions** tab if it is not already selected.

The total exception count, HTTP Error Codes and Error Page Redirects for the selected time range are reported in the upper panel. You can get an exact count of the exceptions per minute at a particular point in time by hovering with your pointing device on the line in the graphs.

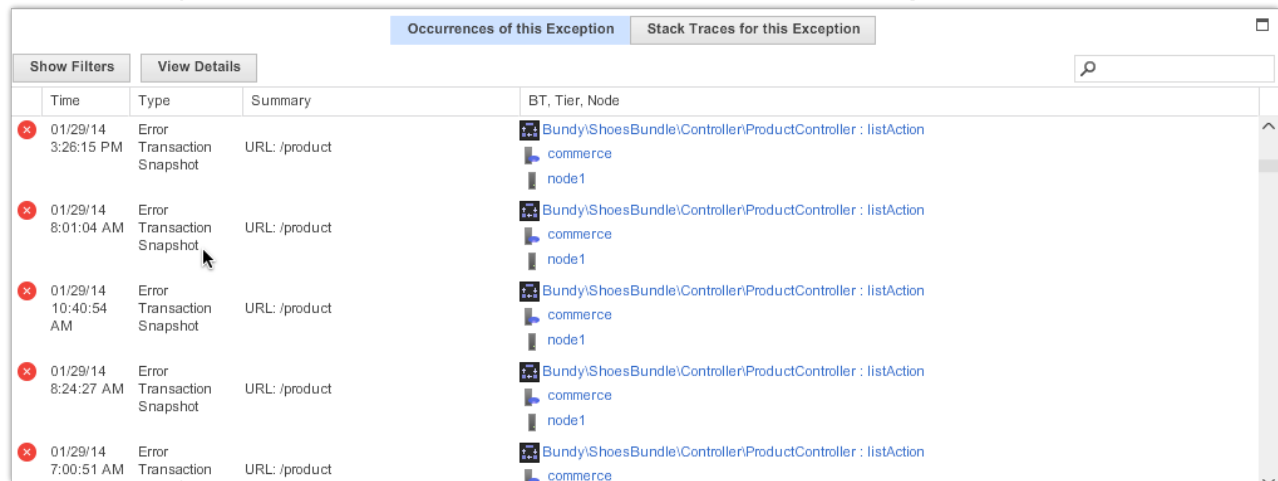
The exceptions list is displayed in the lower panel.

- To filter the exception list, enter the filter term in the filter text field.
- To see only exceptions with performance data, clear the Show Exceptions with 0 count checkbox.

3. To view details of a particular exception, select the exception the list in the lower panel and click **View Details**.

4. To view transaction snapshots for an exception:

- a. In the exception detail window, click the **Occurrences of this Exception** subtab.



Show Filters		View Details		Occurrences of this Exception		Stack Traces for this Exception	
Time	Type	Summary	BT, Tier, Node				
01/29/14 3:26:15 PM	Error Transaction Snapshot	URL: /product	Bundy/ShoesBundle/Controller/ProductController : listAction commerce node1				
01/29/14 8:01:04 AM	Error Transaction Snapshot	URL: /product	Bundy/ShoesBundle/Controller/ProductController : listAction commerce node1				
01/29/14 10:40:54 AM	Error Transaction Snapshot	URL: /product	Bundy/ShoesBundle/Controller/ProductController : listAction commerce node1				
01/29/14 8:24:27 AM	Error Transaction Snapshot	URL: /product	Bundy/ShoesBundle/Controller/ProductController : listAction commerce node1				
01/29/14 7:00:51 AM	Error Transaction Snapshot	URL: /product	Bundy/ShoesBundle/Controller/ProductController : listAction commerce node1				

- b. Select a snapshot from the list.

- c. Click **View Details**.

- d. In the snapshot flow map that displays, click **Drill Down** to get the details of the snapshot. See [Transaction Snapshots](#).

5. To view a stack trace for an exception:

- a. In the exception detail window, click the **Stack Traces for this Exception** tab.

- b. Click an exception in the left panel.

The right panel displays the stack trace for the selected exception.

Occurrences of this Exception		Stack Traces for this Exception
Exception:		Exception:
Exception:		Exception.
Exception:		<pre> at Bundy\ShoesBundle\Controller\ProductController.searchAction(:0) at call_user_func_array(app/bootstrap.php.cache:2913) at Symfony\Component\HttpKernel\HttpKernel.handleRaw(app/bootstrap.php.cache:2885) at Symfony\Component\HttpKernel\HttpKernel.handle(app/bootstrap.php.cache:3024) at Symfony\Component\HttpKernel\DependencyInjection\ContainerAwareHttpKernel.handle(app/bootstrap.php.cache:2305) at Symfony\Component\HttpKernel\HttpKernel.handle(web/app.php:27) at {main}()Controller/ProductController.php:75 </pre>

Learn More

- [Configure Error Detection for PHP](#)
- [Transaction Snapshots](#)

Troubleshoot Node.js Application Problems

Troubleshoot Slow Response Times for Node.js

- [Slow and Stalled Transactions](#)
 - [To troubleshoot slow and stalled transactions](#)
- [High CPU Times](#)
 - [To troubleshoot slow processes](#)
- [Learn More](#)

When you click **Troubleshoot -> Slow Response Times**, the Slow Response Times window opens showing two tabs. You can drill down into transaction issues in the [Slow Transactions](#) tab and into database or remote services issues in the [Slowest DB & Remote Services](#) tab.

Slow and Stalled Transactions

There are many reasons why a business transaction may be slow or stalled.

By default AppDynamics considers a slow transaction one that lasts longer than 3 times the standard deviation for the last two hours and a very slow transaction 4 times the baseline for the last two hours.

By default AppDynamics considers a transaction that lasts longer than 45 seconds (4500 milliseconds) to be stalled.

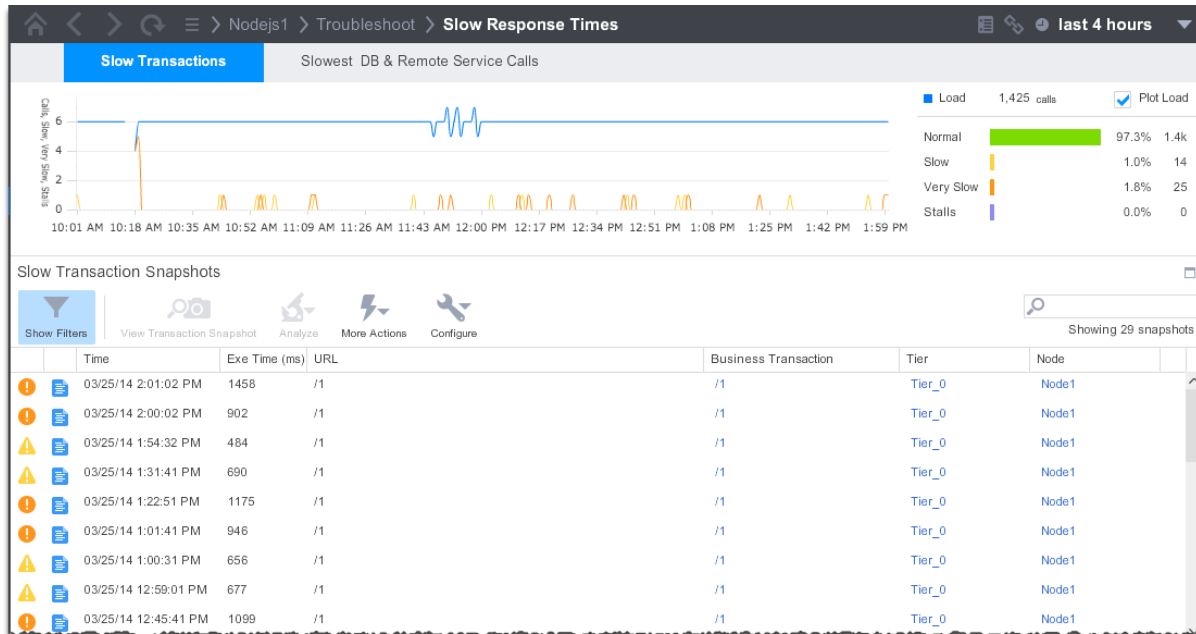
You can configure these thresholds to better match your environment. See [Thresholds](#) and [Configure Thresholds](#).

To troubleshoot slow and stalled transactions

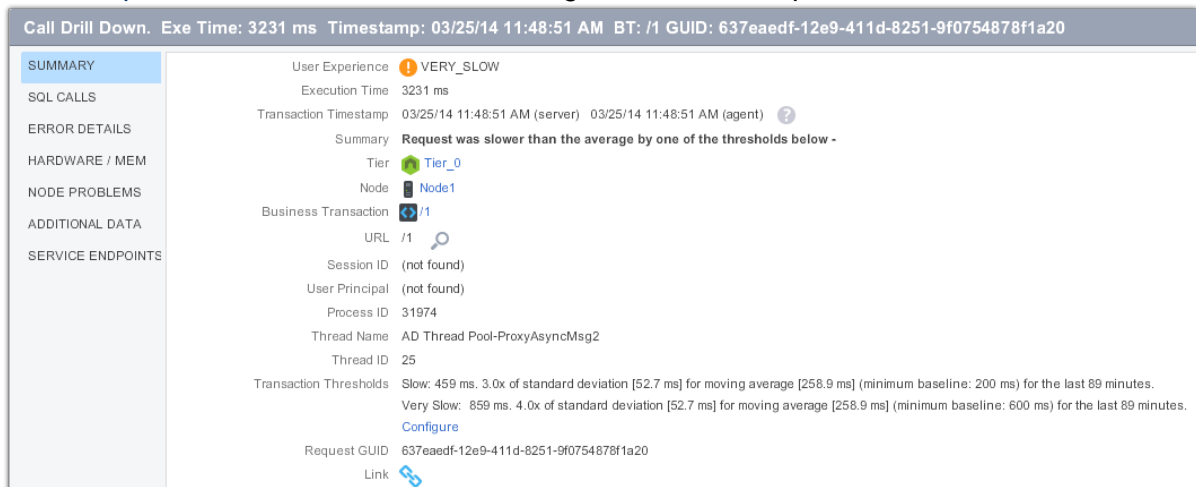
1. Click **Troubleshoot -> Slow Response Times**.
2. Click the **Slow Transactions** tab if it is not selected.
In the upper pane AppDynamics displays a graph of the slow, very slow, and stalled transactions for the time period specified in the Time Range drop-down menu. Click the Plot Load check box to see the load.

In the lower pane AppDynamics displays the transaction snapshots for slow, very slow, and

stalled transactions.



- Click the Exe Time column to sort the transactions from slowest to fastest.
- To drill down, select a snapshot from the list and click **View Transaction Snapshot**. A transaction snapshot shows the details of an instance of a business transaction. See [Transaction Snapshots](#) for information about drilling down into a snapshot.




High CPU Times

Your user experience may be slow because of processes that consume a lot of CPU time blocking other processes.

To troubleshoot slow processes

- In the node dashboard click the Process Snapshots tab.
- Click Collect Process Snapshots.
- In the Collect a Process Snapshot window set the duration to between 1 and 60 seconds.
- Click Start.

In a few minutes you should see some process snapshots in the list.

5. Sort the list by toggling the Exe Time column in the process snapshots list so that the snapshots for the slowest processes are at the top of the list.
6. Double-click one of the slow process snapshots.
7. In the process snapshot, click the CALL GRAPH tab if it is not selected.
8. Examine the Time column in the call graph to identify which of the calls are slow.
9. Click the  at the end of the row for the slow call to see details.

Learn More

- [Monitor Node.js Business Transactions](#)
- [Monitor Node.js Processes](#)
- [Transaction Snapshots](#)
- [Configure Thresholds](#)

Troubleshoot Mobile Applications

Troubleshoot Slow Network Requests from Mobile Applications

- [Identifying the Slowest Network Requests](#)
 - [To identify slow network requests](#)
- [Finding Causes of Slow Network Requests](#)
 - [To investigate details of the slowest individual requests](#)
- [Learn More](#)

Identifying the Slowest Network Requests

First identify which network requests are the slowest.

To identify slow network requests

1. In the left navigation pane click either **End User Experience->iOS** for iOS applications or **End User Experience->Android** for Android applications.
The Mobile APM dashboard opens.
2. Click the Network Requests tab.
3. Click the top of the Network Request Time (ms) column, then toggle it to sort the network requests with the slowest ones at the top.
4. Skip over network requests that you expect to run for a long time or that have very little load (low Requests per Minute).
5. Select and double-click one of the slow network requests that you want to investigate.
6. In the network request dashboard, view the Key Performance Times at the top of the Network Request Dashboard.
If the graph shows that most of the time to service the request was server time, scroll down to the Related Business Transactions section to investigate related business transactions on the server side.
If most of the time is in the network, the request or response body may be too large and is taking a while to transmit. Or the data connection might be slow.

Finding Causes of Slow Network Requests

After you have identified a slow network request that you want to troubleshoot, investigate some individual instances of that network request using network request snapshots.

To investigate details of the slowest individual requests

1. Still in the the Mobile APM dashboard, click the Network Request Snapshots tab.
The Network Request Snapshots List opens.
2. Click **Filters**.
3. In the Network Request Names dropdown list under Network in the Filters panel, check the check box for the network request that you identified in [To identify slow network requests](#), then click **Search**.
This restricts the list to snapshots for that network request only.
4. Click **Filters** again to close the filters panel.
5. In the list, click the top of the Network Request Time (ms) column, then toggle it to sort the network request snapshots with the slowest requests at the top.
6. Select and double-click one of the slow network requests.
The network request snapshot displays the details of the slow request.
7. Scroll down to see if transaction snapshots associated with this network request snapshot are available on the server side.
If transaction snapshots are available and if most of the time for this network request is spent on the server, click on some of the related transaction snapshots to drill down into causes of slow performance on the server. See [Transaction Snapshots](#).

Learn More

- [Monitor Network Requests](#)

Troubleshoot Mobile Application Crashes

- [Identifying Criteria of Applications that Crash Most Often](#)
 - [To identify criteria of applications that crash](#)
- [Finding Causes of Crashes](#)
 - [To find root cause of individual crashes](#)
- [Learn More](#)

Use crash dashboards and crash snapshots to troubleshoot mobile application crashes.

Identifying Criteria of Applications that Crash Most Often

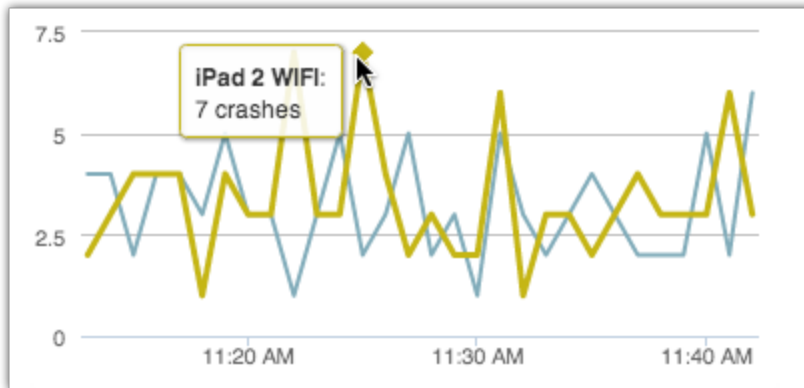
Sometimes most of your crashes share one or more criteria. In other words, your application crashes more often on certain devices or operating systems or carriers or connections.

To identify criteria of applications that crash

1. In the left navigation pane click either **End User Experience->iOS** for iOS applications or **End User Experience->Android** for Android applications.
The Mobile APM dashboard opens.
2. Click the Crashes tab.
3. Click the Dashboard subtab if it is not already selected.
4. In the Crashes vs Requests graph, identify values that are significantly above the Average

line. For example, if a device name is above the line, that type of device is experiencing more crashes than would be expected for the current load.

5. Scroll down to the section for the criteria that seem to be experiencing more crashes.
6. In the line graph, note the times that crashes spike. You can hover over a point on the graph to see the exact number of crashes at that time.



7. Note the criteria (in this example the iPad 2 WIFI device) and the time that most crashes seem to occur.

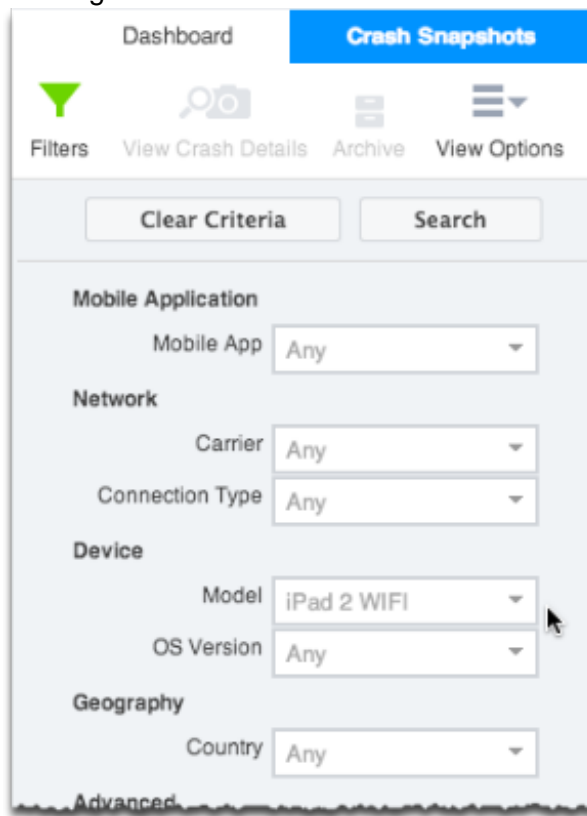
Finding Causes of Crashes

After you have identified which applications are causing most of your crashes and approximately when most crashes occur, you can examine a few of those individual crashes to identify the cause.

To find root cause of individual crashes

1. In the Crashes tab of Mobile APM Dashboard, click the Crash Snapshots subtab.
2. Click **Filters**.
3. Check the check box for the criteria of apps experiencing the most crashes that you identified in [Identifying Criteria of Applications that Crash Most Often](#).

This filters the crash snapshots list to display only snapshots of crashed applications meeting those criteria.



The screenshot shows the 'Crash Snapshots' filter interface in AppDynamics. At the top, there are tabs for 'Dashboard' and 'Crash Snapshots', with 'Crash Snapshots' being the active tab. Below the tabs are four icons: a green funnel for 'Filters', a magnifying glass for 'View Crash Details', a document icon for 'Archive', and a list icon for 'View Options'. Under these icons are two buttons: 'Clear Criteria' and 'Search'. The main section contains several filter categories with dropdown menus: 'Mobile Application' with 'Mobile App' set to 'Any'; 'Network' with 'Carrier' and 'Connection Type' both set to 'Any'; 'Device' with 'Model' set to 'iPad 2 WIFI' and 'OS Version' set to 'Any'; and 'Geography' with 'Country' set to 'Any'. At the bottom left, there is a link labeled 'Advanced'.

4. In the crash snapshots list, select and double-click a snapshot that occurred around the time that most crashes occurred. In the stack trace of the crash snapshot, note the thread and function in which the crash occurred. For some crashes the crashed line number is also available.
5. Optional: Click **Download** to get a text version of the stack trace to send to your application development team.

Learn More

- [Crash Dashboard](#)
- [Crash Snapshots List](#)
- [Crash Snapshots](#)