



# Introduction and Tutorials

AppDynamics Pro Documentation

Version 3.8.x

1. Features Overview .....	3
2. Logical Model .....	7
2.1 Hierarchical Configuration Model .....	9
2.2 Mapping Application Services to the AppDynamics Model .....	10
3. Getting Started .....	13
3.1 Get Started with AppDynamics SaaS .....	14
3.1.1 Use a SaaS Controller .....	18
3.1.2 SaaS Availability and Security .....	19
3.2 Get Started with AppDynamics On-Premise .....	20
3.3 Download AppDynamics Software .....	23
3.4 Quick Start for DevOps .....	26
3.5 Quick Start for Architects .....	27
3.6 Quick Start for Administrators .....	28
3.7 Quick Start for Operators .....	29
3.8 Set User Preferences .....	29
4. Glossary .....	31
5. Tutorials for Java .....	41
5.1 Overview Tutorials for Java .....	41
5.1.1 Use AppDynamics for the First Time with Java .....	42
5.2 Monitoring Tutorials for Java .....	45
5.2.1 Tutorial for Java - Events .....	45
5.2.2 Tutorial for Java - Flow Maps .....	47
5.2.3 Tutorial for Java - Server Health .....	51
5.2.4 Tutorial for Java - Transaction Scorecards .....	55
5.3 Troubleshooting Tutorials for Java .....	58
5.3.1 Tutorial for Java - Business Transaction Health Drilldown .....	58
5.3.2 Tutorial for Java - Exceptions .....	58
5.3.3 Tutorial for Java - Slow Transactions .....	64
5.3.4 Tutorial for Java - Troubleshooting using Events .....	67
6. Tutorials for .NET .....	74
6.1 Overview Tutorials for .NET .....	75
7. Tutorials for PHP .....	76
7.1 First Time Using the App Agent for PHP .....	76
7.2 Tutorial for PHP - Flow Maps .....	79
7.3 Tutorial for PHP - Server Health .....	84
7.4 Tutorial for PHP - Transaction Scorecards .....	87

# Features Overview

This topic describes high-level benefits and features of AppDynamics Pro.

## Expert Advice

### How monitoring analytics can make DevOps more agile

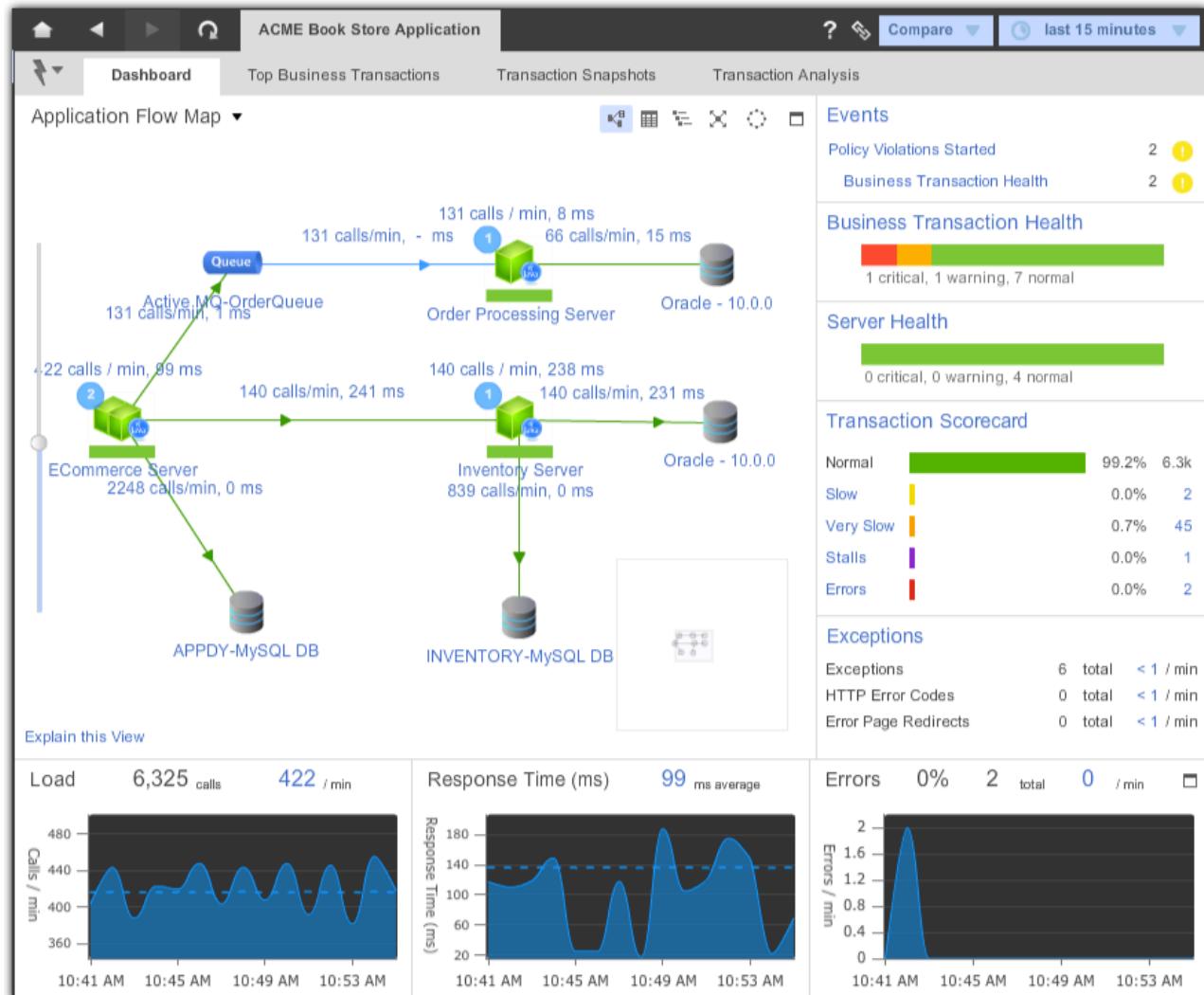
by Sandy Mappic

- Continuous Discovery, Visibility, and Problem Detection
- Real-Time Business Transaction Monitoring
- End User Monitoring
- Service Endpoint Monitoring
- Hardware and Server Monitoring
- Health Rules, Policies, and Actions
- Troubleshooting and Diagnostics
- Systems Integration
- Learn More

## Continuous Discovery, Visibility, and Problem Detection

AppDynamics continuously discovers and monitors all processing in your application environment using advanced tag, trace, and learn technology across your distributed transactions. With this information, AppDynamics provides a simple intuitive view of live application traffic and you can see where bottlenecks exist.

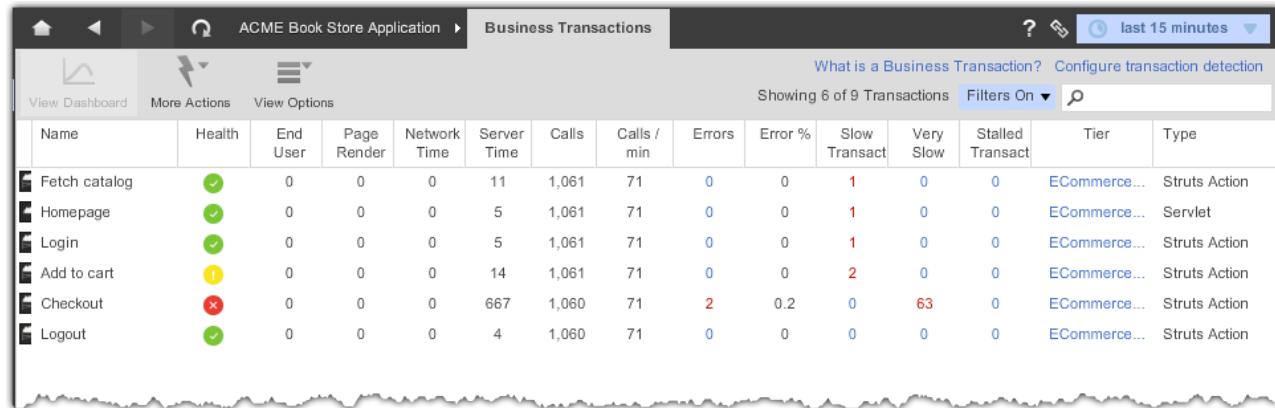
Dashboards show the health of your entire business application. Health indicators are based on configurable thresholds and they update based on live traffic. When new services are added to the system AppDynamics discovers them and adds them to the dashboards and flow maps. See [Visualize App Performance](#).



AppDynamics observes normal performance patterns so that it knows when application performance becomes abnormal. It automatically identifies metrics whose current values are out of the normal range, based on dynamic baselines it has observed for these metrics. See [Behavior Learning and Anomaly Detection](#).

## Real-Time Business Transaction Monitoring

An AppDynamics business transaction represents a distinct logical user activity such as logging in, searching for items, buying an item, etc. Organizing application traffic into business transactions aligns the traffic with the primary functions of a web business. This approach focuses on how your users are experiencing the site and provides real-time performance monitoring.

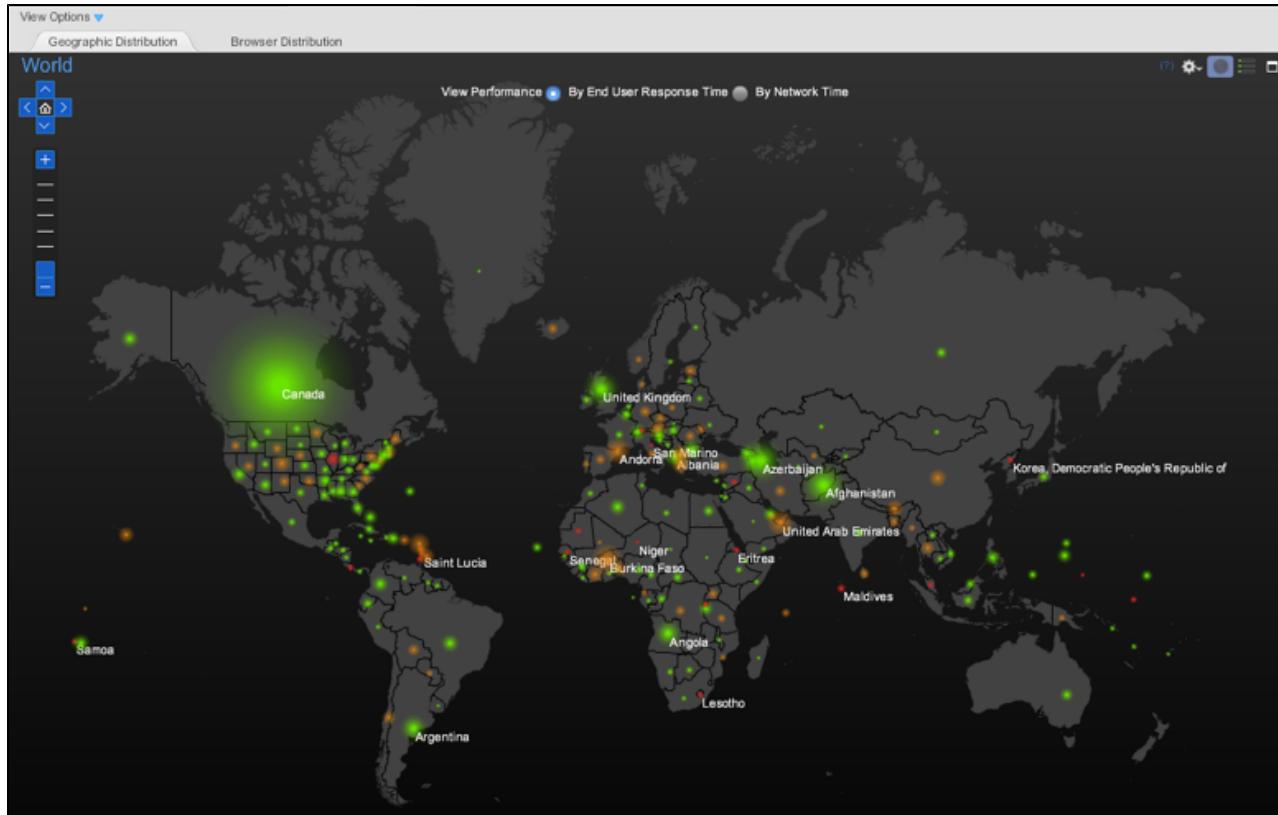


See [Business Transaction Monitoring](#) and [Background Task Monitoring](#).

## End User Monitoring

End user monitoring (EUM) provides information about your end users' experience starting from the users' web browsers and their native mobile applications. It gives you visibility across geographies and browser types, answering questions such as:

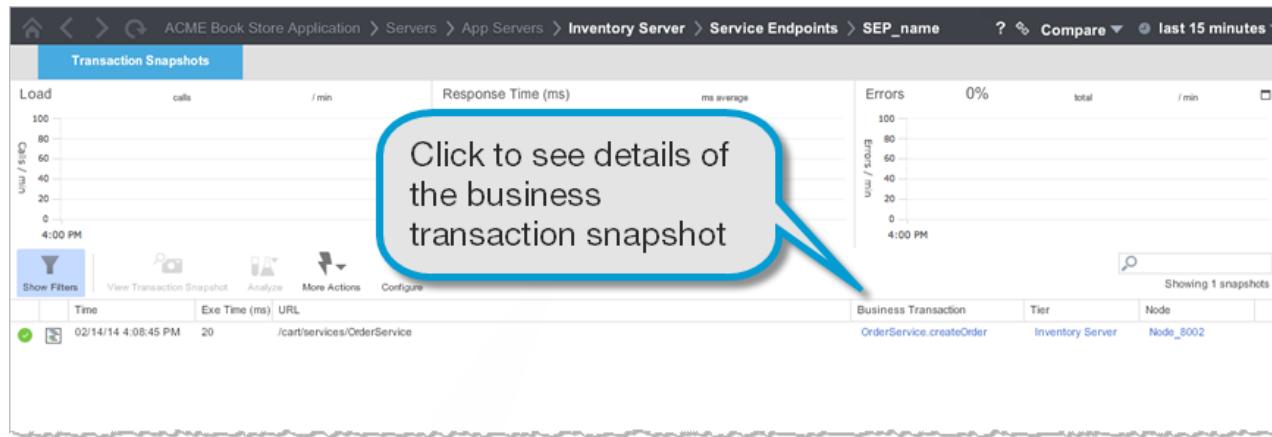
- Where are the heaviest loads?
- Where are the slowest end-user response times?
- How does end user performance vary by Web browser?
- How does end user performance vary by mobile application, carrier, or device?



See AppDynamics End User Experience.

## Service Endpoint Monitoring

Service endpoints are helpful in complex, large-scale applications where an owner is assigned to one or more logical tiers and the standard representation does not correspond with real-life ownership of application components. Service endpoints allow you to see a subset of the metrics for the tier so you can focus on the key performance indicators and snapshots of entry points that are truly of interest to you. Service endpoints are similar to business transactions except that they only show metrics for the entry points and do not track metrics for any downstream segments.



See Service Endpoint Monitoring.

## Hardware and Server Monitoring

AppDynamics machine agents gather information about the operating systems and machines, such as CPU activity, memory usage, disk reads and writes, etc. AppDynamics agents monitor JVM and CLR metrics including heap usage and collections. See [Infrastructure Monitoring](#).

## Health Rules, Policies, and Actions

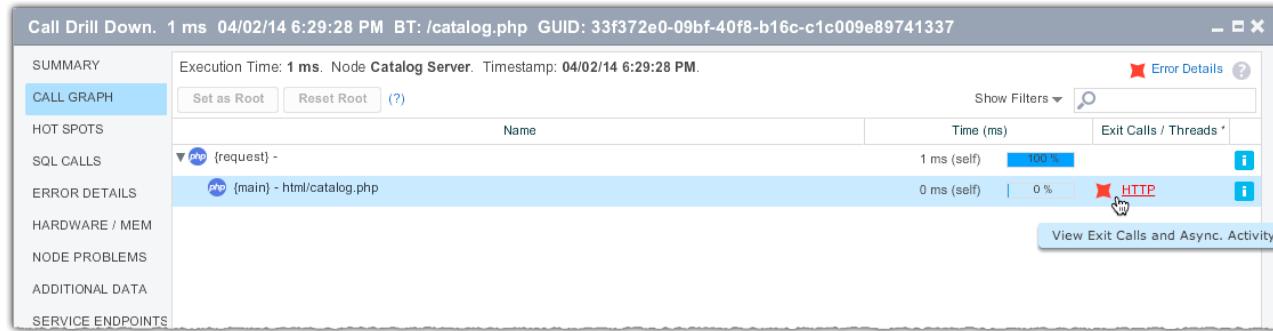
Dynamic baselines combined with policies and health rules help you proactively detect and troubleshoot problems before customers are affected. Health rules define metric conditions to monitor, such as when the "average response time is four times slower than the baseline". AppDynamics supplies default health rules that you can customize, and you can create new ones.

You can configure policies to trigger automatic actions when a health rule is violated or when any event occurs. Actions include sending email, scaling-up capacity in a cloud or virtualized environment, taking a thread dump, or running a local script. See [Alert and Respond](#).

## Troubleshooting and Diagnostics

You can examine transaction snapshots for slow and error transactions and drill down into the snapshot with the slowest response time to begin deep diagnostics to discover the root cause of

the problem.



See [Rapid Troubleshooting](#).

## Systems Integration

AppDynamics is designed to interface with other systems in your organization. You can add data to AppDynamics, retrieve data from AppDynamics, and integrate AppDynamics actions into your alerting system. See [AppDynamics Extensions and Integrations](#).

## Learn More

- [Product Features and Benefits](#)

## Logical Model

- [Business Application](#)
- [Tiers](#)
- [Nodes](#)
- [Learn More](#)

This topic describes the basic elements of the AppDynamics model.

Before deploying AppDynamics, also see [Mapping Application Services to the AppDynamics Model](#).

## Business Application

An AppDynamics business application models all components or modules in an application environment that provide a complete set of functionality. Think of it as all the web applications, databases, and services that interact or "talk" to each other or to a shared component. When web applications, databases, and services interact, AppDynamics can correlate their activities to provide useful and interesting performance data.

AppDynamics lets you monitor multiple business applications, though it does not correlate events between them.

Because a single node belongs to a single business application, you can also think of a business application as a kind of namespace for all your nodes. See [Nodes](#).

Business applications contain tiers, and tiers contain nodes.

## Tiers

A tier represents a key module in an application environment, such as a website or processing application or a virtual machine. Tiers help you logically organize and manage your business application so that you can scale multiple nodes, partition metrics, define performance thresholds, and respond to anomalies. The metrics from one tier tell a different story than those from another tier; AppDynamics helps you define different policies and processes for each tier. A tier can belong to only one business application.

A tier is composed of one node or a group of nodes. For example, in the Acme sample application the Inventory tier has one node whereas the E-Commerce tier has 2 nodes.

Nodes grouped into a tier may have redundant functionality or may not. An example of a multi-node tier with redundant nodes is when you have a set of clustered application servers or services. An example of a multi-node tier with different nodes is when you have a set of services that do not interact with each other though you want to roll up their performance metrics together.

Keep in mind that an environment can have similar nodes that are used by different applications, so similar nodes should not always belong to the same tier. An example is a complex environment that has two HTTP web servers that serve two separate applications.

Business applications contain tiers. The traffic in a business application flows between tiers. This flow is represented in AppDynamics flow maps along with performance data for the traffic. There is always a tier that is the starting point for a Business Transaction, indicated by a Start label on the flow map.

## Nodes

A node is the basic unit of processing that AppDynamics monitors. By definition a node is instrumented by an AppDynamics agent, either an app agent or machine agent or both. App agents are installed on:

- JVMs
- Windows .NET applications (IIS, executables, or services)
- PHP Runtime Instances
- Node.js processes

Machine agents are installed on virtual or physical machine operating systems.

Nodes belong to tiers. An app agent node cannot belong to more than one tier. A machine agent cannot belong to more than one tier; however you can install more than one machine agent on a machine.

## Learn More

- [Mapping Application Services to the AppDynamics Model](#)
- [Name Business Applications, Tiers, and Nodes](#)
- [Features Overview](#)

- Glossary

## Hierarchical Configuration Model

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

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

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

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

## Entry Point and Exit Point Inheritance

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

The screenshot shows the 'Entry Points' configuration for the 'Acme Online Book Store' application. The 'ECommerce-Server' tier is selected. The 'Java - Transaction Detection' tab is active. Buttons for 'Copy' and 'Configure all Tiers to use this Configuration' are visible.

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

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

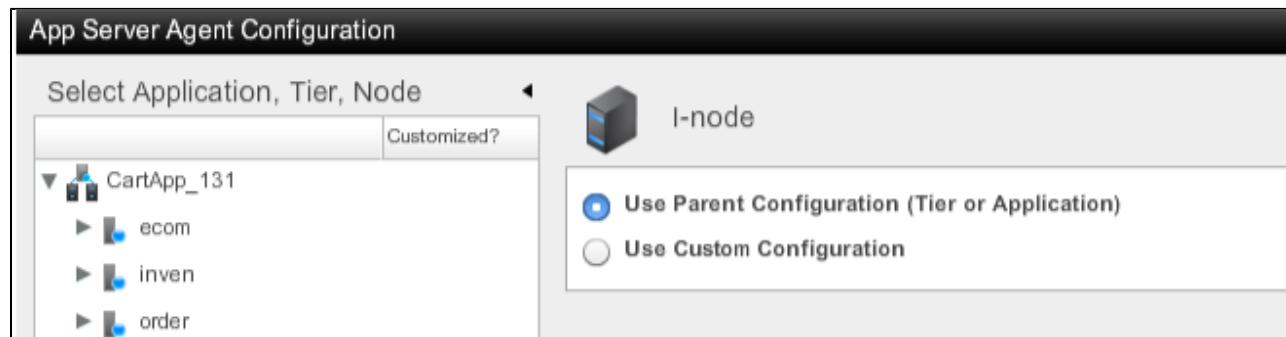
You can configure all tiers to use the custom configuration or copy the configuration for re-use in specific tiers. You can also reset a tier that is currently using a custom configuration to use the application-level configuration.

The screenshot shows the configuration for the 'ECommerce-Server' tier. The 'Use Custom Configuration for this Tier' radio button is selected. Buttons for 'Copy', 'Reset to Application Configuration', and 'Configure all Tiers to use this Configuration' are visible.

## Node Inheritance

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

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



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

## Switching Configuration Levels

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

## Learn More

- Configure Backend Detection (Java)
- Configure Backend Detection (.NET)
- Configure Business Transaction Detection
- App Agent Node Properties

## Mapping Application Services to the AppDynamics Model

- Your Application and the AppDynamics Model
- How AppDynamics Represents Your Application
  - AppDynamics Business Applications
  - AppDynamics Tiers
- How to Map
- Learn More

AppDynamics and your team may use different terminology to describe your application environment. This topic discusses how to map the services in your application environment to the AppDynamics model, which uses the terms "business applications", "tiers", and "nodes". For an overview of these terms see [Logical Model](#).

## Your Application and the AppDynamics Model

Your distributed application environment most likely consists of various services, including:

- Web applications served from an application server (JVM, IIS, PHP Web server, etc.)

- Databases or other data stores
- Remote services such as message queues and caches

AppDynamics maps your application environment into a hierarchical system of business applications, tiers, nodes and backends.

The node represents the actual application server that is instrumented by an AppDynamics app agent.

Business applications and tiers are logical constructs used to represent your environment in the AppDynamics model.

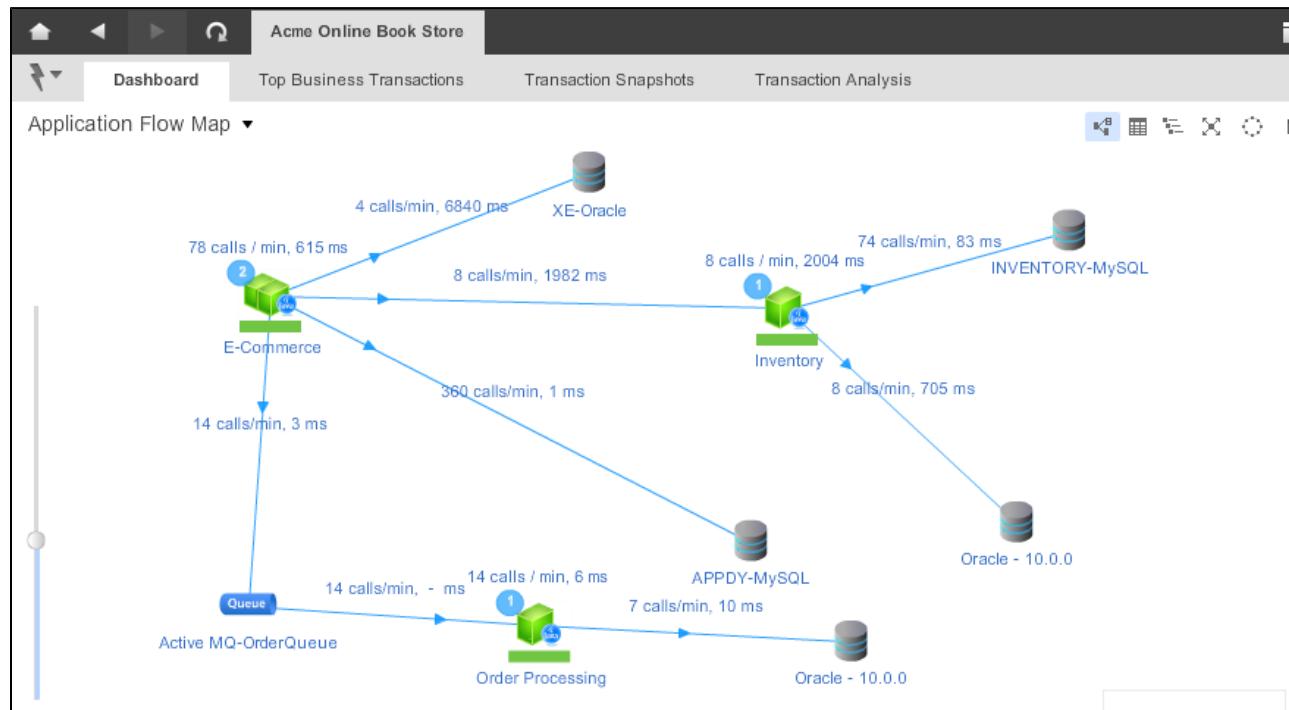
Business applications contain tiers and tiers contain nodes. A node cannot belong to more than one tier, and a tier cannot belong to more than one business application.

A backend is a component that is not instrumented by an AppDynamics app agent, but the model allows you to monitor the flows from the instrumented nodes to the backends. These flows often reveal the root cause of a problem that is first identified on an instrumented node.

## How AppDynamics Represents Your Application

The flowmap below describes a single business application for the Acme Online Book Store.

- E-Commerce, Order Processing and Inventory are the tiers.
- The boxes inside the tiers represent instrumented nodes. The E-Commerce tier has two nodes, the Order Processing and Inventory tiers each has one node.
- The database backends are XE-Oracle, Inventory-MySQL, APPDY-MySQL and two Oracle 10.0.0s.
- The message queue backend is Active MQ--Order Queue.
- The blue lines represent the flow of traffic through the entire application.



Because the services provided by the E-Commerce, Inventory, and Order Processing interacting tiers are all modeled as part of the same business application, it is possible, for example, for AppDynamics to trace the root cause of poor performance of a node in the front-end E-Commerce tier to a slow SQL call from the downstream Inventory tier to the INVENTORY-MySQL database. Without this correlation among the services, this information would not be available.

## AppDynamics Business Applications

You can use a single AppDynamics business application to model all of the application environment's services that provide a complete set of functionality. Think of the business application as all the services that interact to support the application's mission. When these services (web applications, databases, remote services, etc.) interact, they are modeled as part of the same business application, and AppDynamics can correlate performance metrics among them to provide a complete picture of the application's performance. A complete picture helps you identify the root cause of any problems that are detected. If any of the services upon which the application depends are missing from the model, you may miss information about a component that is causing problems to appear in a different component. AppDynamics cannot provide correlation between separate business applications.

For example, a single shopping business application may be composed of an inventory application, a e-commerce front-end application, and databases. The inventory application and e-commerce front-end application could be modeled as tiers in a single AppDynamics "business application".

On the other hand, if you do not care about correlation among these services and instead want to maintain separate access control to the various components, you could model the services as separate business applications.

It is also appropriate to have multiple business applications for sets of services that do not interact with each other. A typical example of using multiple business applications is when you have separate staging, testing, and production environments for the same website. In this case the three business applications are essentially copies of each other.

## AppDynamics Tiers

An AppDynamics tier represents an instrumented service (such as a web application) or multiple services that perform the exact same functionality and may even run the same code. These services may be thought of as "applications" in your application environment, but if they interact with one another AppDynamics usually models them as tiers in the same "business application".

A tier can be composed of one node or multiple redundant nodes. One example of a multi-node tier is a set of clustered application servers or services.

There is no interaction among nodes within a single tier. Interaction occurs between tiers in a business application, as illustrated in the [flowmap](#).

## How to Map

The mapping of tiers to business applications and of nodes to tiers occurs in the configuration of the app agent, either in the options to the app agent startup script or in the controller-info-xml file.

For example, in an all-Java environment, to map Node\_8000 and Node\_8003 to the E-Commerce

tier in the AcmeOnLine business application, the startup options for Node\_8000 would be

```
-Dappdynamics.agent.applicationName=AcmeOnLine  
-Dappdynamics.agent.tierName=E-Commerce  
-Dappdynamics.agent.nodeName=Node_8000
```

and for Node\_8003

```
-Dappdynamics.agent.applicationName=AcmeOnLine  
-Dappdynamics.agent.tierName=E-Commerce  
-Dappdynamics.agent.nodeName=Node_8003
```

To map Node\_8002 in the Inventory tier in the same business application, the configuration would be

```
-Dappdynamics.agent.applicationName=AcmeOnLine  
-Dappdynamics.agent.tierName=Inventory  
-Dappdynamics.agent.nodeName=Node_8002
```

and to map Node\_8001 in the Order Processing tier in the same application

```
-Dappdynamics.agent.applicationName=AcmeOnLine  
-Dappdynamics.agent.tierName=Order Processing  
-Dappdynamics.agent.nodeName=Node_8001
```

Details vary depending on the platform of the agent. See the installation and configuration properties documentation for the particular app agent that you are configuring.

## Learn More

- [Logical Model](#)
- [Name Business Applications, Tiers, and Nodes](#)
- [App Agent for Java Configuration Properties](#)
- [App Agent for .NET Configuration Properties](#)
- [App Agent for PHP Proxy Configuration Properties](#)
- [Install the App Agent for PHP](#)
- [Install the App Agent for Node.js](#)

## Getting Started

- [Initial Installation](#)
  - [Self-Service Trial or Standard?](#)
  - [On-premise or SaaS](#)
    - [Get Started with AppDynamics SaaS](#)
    - [Get Started With AppDynamics On-Premise](#)

- Monitoring, Troubleshooting, and Analyzing Application Performance

This section gives you a roadmap to using AppDynamics.

## Initial Installation

### Self-Service Trial or Standard?

If you are using the self-service trial see [Quick Install](#).

If you are using a standard installation see [Install and Upgrade AppDynamics](#).

### On-premise or SaaS?

To get started with installing, configuring, and using AppDynamics, first determine whether you will use an [on-premise](#) or [SaaS](#) Controller.

For information about the different approaches see:

- [SaaS Availability and Security](#)
- [Differences when using a SaaS Controller](#)

### Get Started with AppDynamics SaaS

If you are using or going to use the AppDynamics SaaS Controller, see [Get Started with AppDynamics SaaS](#).

### Get Started With AppDynamics On-Premise

If you are going to host your own Controller on premise, see [Get Started With AppDynamics On-Premise](#).

## Monitoring, Troubleshooting, and Analyzing Application Performance

To get started using AppDynamics after it is installed see:

- [AppDynamics Essentials](#)
- [Quick Tour of the User Interface Video Tutorial](#) 

### Get Started with AppDynamics SaaS

Follow these steps to get started with AppDynamics.

If you are reading a PDF of this document, use your Help Center login to access the documentation at <http://docs.appdynamics.com>.

- [Get Your SaaS Account](#)

#### Expert Advice

##### Deploying APM in the Enterprise... the Path of the Rock Star

By Jim Hirschauer

Information from  
AppDynamics

- Design Your AppDynamics Deployment
- Download and Install the AppDynamics App Agents
- Download and Install the AppDynamics Web and Mobile Agents
- SaaS Login Credentials
- Connecting Agents to Your SaaS Controller Service
- Access the AppDynamics UI from a Browser
- Review the Dashboards and Flow Maps
- Review Defaults and Configure Business Transactions, if Needed
- Review Defaults and Configure Client-Side Monitoring, if Needed
- Review Defaults and Configure Databases and Remote Services, if Needed
- Review Default Health Rules and Set Up Policies
- Review Default Error Detection
- Explore Additional Data and Metric Features
- Configure Advanced Features
- Start Monitoring and Troubleshooting
- Questions?

## Get Your SaaS Account Information from AppDynamics

After signing up for AppDynamics SaaS, you receive a Welcome email containing important account information, including the [Account Owner](#) login. Save this information.

## Design Your AppDynamics Deployment

- Learn about [Business Transaction Monitoring](#) and identify which critical business transactions you want to monitor.
- Learn about [AppDynamics End User Experience](#) and decide whether you want to use this feature.
- Learn about how to map your application components to the AppDynamics business application, tier, and node model. See [Logical Model](#) and [Name Business Applications, Tiers, and Nodes](#).
- Based on the model, plan how you will specify AppDynamics application, tier, and node names during installation.
- Decide whether you want to monitor client-side usage with [AppDynamics End User Experience](#).
- For Java environments, decide whether you want to use [object instance tracking](#).

## Download and Install the AppDynamics App Agents

Download the AppDynamics application agents from the [Download Center](#). AppDynamics app agents collect data from your application servers and other monitored systems and report to the Controller. Select the agents that are appropriate for your environment:

- Java Agent
- .NET Agent
- PHP Agent
- Machine Agent

For details see [Download AppDynamics Software](#).

Follow the [instructions](#) to install the AppDynamics App Agents.

## Download and Install the AppDynamics Web and Mobile Agents

Install the client-side agents in your mobile applications and web pages. See [instructions for mobile](#) and [web](#).

## SaaS Login Credentials

SaaS Controller login credentials are included in the welcome email from AppDynamics.

To add additional login accounts contact the [AppDynamics Support Team](#).

The SaaS Controller login is an Account Administrator credential. The Account Administrator can create other users for the account. See [Account Administrator](#).

## Connecting Agents to Your SaaS Controller Service

For agents to successfully connect to the Controller, configure the Controller host and port information using either the controller-info.xml file or the system properties of your JVM startup script.

To use HTTPS communication, enable SSL by setting the <controller-ssl-enabled> agent configuration property to "True". For details see [App Agent for Java Configuration Properties](#), [App Agent for .NET Configuration Properties](#), [App Agent for PHP Proxy Configuration Properties](#) and [Machine Agent Configuration Properties](#).

- The default ports for the SaaS Controller service are:
  - Port 80 for HTTP
  - Port 443 for HTTPS

If you need to specifically open up the communication ports (80 or 443) for the AppDynamics SaaS Controller IP address please request the IPs from the [AppDynamics Support Team](#).

## Access the AppDynamics UI from a Browser

Once you have installed the agents, launch your web browser and connect to the AppDynamics User Interface (UI). For SaaS, the URL includes the account name from the Welcome email:

http://<account-name>.saas.appdynamics.com/controller

When using SSL, use port 443 or https to access the Controller.

## Review the Dashboards and Flow Maps

AppDynamics automatically discovers the [Business Transactions](#) in your application environment. Browse the [Application Dashboard](#) and see the [Flow Maps](#) to visualize your application. You can resize and move icons around on the flow maps.

## Review Defaults and Configure Business Transactions, if Needed

The default configurations may need to be further customized for your environment. For example, AppDynamics may have discovered transactions that you want to group together or even exclude, because you want to concentrate on the most important transactions. There may be business transactions that are not yet discovered for which you need to configure detection rules. See:

- [Monitor Business Transactions](#)
- [Configure Business Transaction Detection](#)

## **Review Defaults and Configure Client-Side Monitoring, if Needed**

You may want to refine the way AppDynamics names pages and mobile requests, for example, if the data for multiple web pages would be better understood under a single name. See:

- [Configure Mobile Network Requests](#)
- [Set Up and Configure Web EUM](#)

## **Review Defaults and Configure Databases and Remote Services, if Needed**

AppDynamics automatically discovers "backends" such as databases, message queues, etc. by following calls in the Java or .NET code. Look at the [databases](#) and [remote services](#) dashboards to make sure all necessary backends are revealed. If needed, configure how backends are detected.

## **Review Default Health Rules and Set Up Policies**

AppDynamics provides default [Health Rules](#) that define performance parameters for business transactions, such as the conditions that indicate a slow transaction, or when too much memory is being used. You can adjust the thresholds that define when a health rule is violated, create new health rules, and [set up policies](#) to specify actions to automate when health rules are violated.

## **Review Default Error Detection**

AppDynamics detects errors and exceptions. You can review and, if needed, modify the [error detection rules](#). For example, some errors you may want to ignore.

## **Explore Additional Data and Metric Features**

Explore these features to gain more insight into application performance:

- [Data Collectors](#)
- [Business Metrics](#)
- (for Java environments) [JMX Metrics](#)
- [Machine Agent Custom Metrics](#)

## **Configure Advanced Features**

Additional features you may want to use include:

- [Custom Dashboards](#)
- [Automation](#)
- [AppDynamics Extensions and Integrations](#)

## Start Monitoring and Troubleshooting

Start getting the benefits of AppDynamics! See:

- [AppDynamics in Action Videos](#)
- [AppDynamics Features](#)

## Questions?

For questions about using AppDynamics contact the [AppDynamics Support Team](#).

## Use a SaaS Controller

- Your SaaS Controller URL
- Login Credentials
- Connecting Agents to Your SaaS Controller Service
- SMTP Service for SaaS
- Contact Support

If you are using the SaaS service for the AppDynamics Controller, simply open a web browser at the URL of the AppDynamics UI and log in with your AppDynamics credentials.

## Your SaaS Controller URL

Your SaaS Controller URL is included in the welcome email from AppDynamics.

The URL is of the following form:

```
http(s)://<customer>.saas.appdynamics.com/controller
```

## Login Credentials

Login credentials are included in the welcome email from AppDynamics.

To add additional login accounts contact the [AppDynamics Support Team](#).

## Connecting Agents to Your SaaS Controller Service

For agents to successfully connect to the Controller, configure the Controller host and port information using either the controller-info.xml file or the system properties of your JVM startup script.

To use HTTPS communication, enable SSL by setting the <controller-ssl-enabled> agent configuration property to "True". For details see [App Agent for Java Configuration Properties](#), [App Agent for .NET Configuration Properties](#), and [Machine Agent Configuration Properties](#). See also [Implement SSL on SaaS](#).

- The default ports for the SaaS Controller service are:
  - Port 80 for HTTP
  - Port 443 for HTTPS

 **Important** If you need to specifically open up the communication ports (80 or 443) for the AppDynamics SaaS Controller IP address the subnet range is: 69.27.44.0/24.

## SMTP Service for SaaS

To enable email and SMS notifications you must configure SMTP. See [Configure the SMTP Server](#).

For SaaS users, AppDynamics has an SMTP service running on every machine.

The configuration is:

**SMTP Host:** localhost

**SMTP Port:** 25

No authentication is needed.

## Contact Support

For questions about the service contact the [AppDynamics Support Team](#).

## SaaS Availability and Security

- Service Availability
- Customer Account Login Security
- Hosting
- Data Access
- Data Collection
- Data Communication

This topic summarizes the service availability and security AppDynamics provides for customers who use the AppDynamics SaaS platform.

### Service Availability

AppDynamics makes every best effort to operate and manage the AppDynamics SaaS platform with a goal of 99.5% uptime Service Level Agreement (SLA), excluding planned maintenance windows. AppDynamics actively monitors the latency of the SaaS platform 24/7 from different locations around the world to ensure AppDynamics delivers the best quality of service.

### Customer Account Login Security

The AppDynamics user interface (UI) uses TLS 1.0 with AES 256 bit encryption terminated at the server to ensure end-to-end security over the wire.

For additional security, AppDynamics can restrict UI access to customer corporate networks. This is available for dedicated SaaS hosting plans only.

### Hosting

The AppDynamics SaaS platform (servers, infrastructure and storage) is hosted in one of the largest Tier III data centers in North America. The data center is designed and constructed to deliver world-class physical security, power availability, infrastructure flexibility, and growth capacity. The data center provider is SSAE 16 SOC 1 Type II compliant, which means that it has been fully independently audited to verify the validity and functionality of its control activities and processes.

Every server is operated in a fully redundant fail-over pair to ensure high availability. Data is

backed up nightly, stored redundantly and can be restored rapidly in case of failure. AppDynamics also provides an off-site backup service that is available at additional cost.

Security updates and patches are actively evaluated by engineers and are deployed based upon the security risks and stability benefits they offer to the AppDynamics SaaS platform and customers.

#### Data Access

Access to the AppDynamics SaaS platform infrastructure and data is secured by multiple authentication challenges including RSA and DSA key pairs, passwords, and network access control lists. Infrastructure and data access is restricted to AppDynamics employees and contractors, all of whom are under strict confidentiality agreements.

System and Network activity is actively monitored by a team of engineers 24/7. Failed authentication attempts are audited and engineers are paged immediately so that any possible intrusion or threat can be investigated promptly. Standard firewall policies are deployed to block all access except to ports required for AppDynamics SaaS platform and agent communication.

#### Data Collection

AppDynamics agents collect metrics that relate to the performance, health and resources of an application, its components (transactions, code libraries) and related infrastructure (nodes, tiers) that service those components.

#### Data Communication

AppDynamics agents typically push data using one-way HTTP or HTTPS connections to a single host (a Controller) which has been allocated to one or more customer accounts. AppDynamics offers dedicated Controllers for customers who require their data to be isolated.

For added security, agents can be configured to send data using encrypted transmission by simply selecting HTTPS port 443 and setting "controller-ssl-enabled" to true in the agent configuration. AppDynamics agents also have built-in support for outbound HTTP proxies for customers using these security mechanisms.

AppDynamics uses random staggering on agent data communication to the AppDynamics SaaS platform so traffic is spread evenly to minimize bursts and spikes of network traffic from your data center to the AppDynamics SaaS platform.

The following table shows typical bandwidth usage by number of agents, given the default agent configuration and typical application conditions:

Number of Agents	Typical Network Bandwidth Used (per minute)
1	300 Kbit to 500 Kbit
100	30 Mbit to 50 Mbit
1000	300 Mbit to 500 Mbit

These figures assume a 1:1 relationship between an agent and a JVM/CLR.

## Get Started with AppDynamics On-Premise

Follow these steps to get started with AppDynamics.

If you are reading a PDF of this document, use your Help Center login to access additional documentation at <http://docs.appdynamics.com>.

#### Expert Advice

##### [Deploying APM in the Enterprise... the Path of the Rock Star](#)

By Jim Hirschauer

- Design Your AppDynamics Deployment
- Size and Verify the Controller Environment
- Download AppDynamics
- Install the AppDynamics Controller
- Install the AppDynamics App Agents
- Install the AppDynamics Web and Mobile Agents
- Review the Dashboards and Flow Maps
- Review Defaults and Configure Business Transactions, if Needed
- Review Defaults and Configure Client-Side Monitoring, if Needed
- Review Default Health Rules and Set Up Policies
- Review Default Error Detection
- Explore Additional Data and Metric Features
- Configure Advanced Features
- Start Monitoring and Troubleshooting

## Design Your AppDynamics Deployment

- Learn about [Business Transaction Monitoring](#) and identify which critical business transactions you want to monitor.
- Learn about [AppDynamics End User Experience](#) and decide whether you want to use this feature.
- Learn about how to map your application components to the AppDynamics business application, tier, and node model. See [Logical Model](#) and [Name Business Applications, Tiers, and Nodes](#).
- Based on the model, plan how you will specify AppDynamics application, tier, and node names during installation.
- Decide whether you want to monitor client-side usage with [AppDynamics End User Experience](#).
- For Java environments, decide whether you want to use [object instance tracking](#).

## Size and Verify the Controller Environment

- Verify that you have the resources to support system requirements and the Controller performance profile. The profile reflects the number of nodes and AppDynamics applications that the Controller will monitor. For details see [Controller System Requirements](#).

## Download AppDynamics

- Download the AppDynamics software components from the [Download Center](#). For details see [Download AppDynamics Software](#).

## Install the AppDynamics Controller

The AppDynamics Controller is the central management server where all data is stored and analyzed. All AppDynamics Agents connect to the Controller to report data, and the Controller provides a browser-based user interface for monitoring and troubleshooting application performance. A wizard installs the Controller in just a few minutes. Install the AppDynamics Controller only if you are using the on-premise Controller deployment option.

- Follow the [instructions to install an on-premise Controller](#).
- Important installation and configuration considerations include:
  - [High Availability](#)
  - [Backups](#)
  - [SSL and Certificates](#)
  - [User Authentication with LDAP or SAML](#)

## Install the AppDynamics App Agents

AppDynamics Application Agents collect data from your application servers and other monitored systems and report to the Controller. Install them on the application servers you want to instrument and any other machines you want to monitor. Follow the [instructions to install the AppDynamics App Agents](#).

## Install the AppDynamics Web and Mobile Agents

Install the client-side agents in your mobile applications and web pages. See [instructions for mobile](#) and [web](#).

## Access the AppDynamics UI from a Browser

Once you have installed the Controller and agents, launch your web browser and connect to the AppDynamics User Interface (UI).

- For an on-premise Controller, the URL pattern is:

```
http://<controller-host>:<controller-port>/controller
```

When using SSL, use port 443 or https to access the Controller.

## Review the Dashboards and Flow Maps

AppDynamics automatically discovers the [Business Transactions](#) in your application environment. Browse the [Application Dashboard](#) and see the [Flow Maps](#) to visualize your application. You can resize and move icons around on the flow maps.

## Review Defaults and Configure Business Transactions, if Needed

The default configurations may need to be further customized for your environment. For example, AppDynamics may have discovered transactions that you want to group together or even exclude, because you want to concentrate on the most important transactions. There may be business transactions that are not yet discovered for which you need to configure detection rules. See:

- Business Transaction Monitoring
- Configure Business Transaction Detection

## Review Defaults and Configure Client-Side Monitoring, if Needed

You may want to refine the way AppDynamics names pages and mobile requests, for example, if the data for multiple web pages would be better understood under a single name. See:

- Configure Mobile Network Requests
- Set Up and Configure Web EUM

## Review Defaults and Configure Databases and Remote Services, if Needed

AppDynamics automatically discovers "backends" such as databases, message queues, etc. by following calls in the application code. Look at the [databases](#) and [remote services](#) dashboards to make sure all necessary backends are revealed. If needed, [change how backends are detected](#).

## Review Default Health Rules and Set Up Policies

AppDynamics provides default [Health Rules](#) that define performance parameters for business transactions, such as the conditions that indicate a slow transaction, or when too much memory is being used. You can adjust the thresholds that define when a health rules is violated, create new health rules, and [set up policies](#) to specify actions to automate when health rules are violated.

## Review Default Error Detection

AppDynamics detects errors and exceptions. You can review and, if needed, modify the [error detection rules](#). For example, some errors you may want to ignore.

## Explore Additional Data and Metric Features

Explore these features to gain more insight into application performance:

- Data Collectors
- Business Metrics
- (for Java environments) JMX Metrics
- Machine Agent Custom Metrics

## Configure Advanced Features

Additional features you may want to use include:

- Custom Dashboards
- Automation
- AppDynamics Extensions and Integrations

## Start Monitoring and Troubleshooting

Start getting the benefits of AppDynamics! See:

- [AppDynamics in Action Videos](#)
- [AppDynamics Features](#)

## Download AppDynamics Software

- Accessing the AppDynamics Download Center
  - Download Tips
- AppDynamics Software Components
- Access to Older Versions
- Downloading from the Linux Shell
- Learn More

### Accessing the AppDynamics Download Center

You should have received a Welcome email from AppDynamics. The Welcome email contains credentials for you to log in to the [AppDynamics Support Center](#).

If you have not received this Welcome email, contact your AppDynamics Sales Representative or email [support@appdynamics.com](mailto:support@appdynamics.com).

Access the [AppDynamics Download Center](http://download.appdynamics.com) (<http://download.appdynamics.com>) and browse to the appropriate section on the Download Center to download the relevant files.

### Download Tips

Always copy or transfer the downloaded files in binary mode.

If you have downloaded a binary on Windows, and you are moving it to a Unix environment, the transfer program must use binary mode.

 For each file you download, verify that the download is complete and that the file is not corrupted. [Run a checksum tool](#) and compare the results against the checksum information on the download site.

## AppDynamics Software Components

AppDynamics Software Component	Description	SaaS	On-Premise
Controller	Central management server where all data is stored and analyzed.	N/A	Required
Java App Server Agent	Instrumentation Agent for Java virtual machines. This component must be installed on each Java application server you want to instrument through AppDynamics.	Required for Java	Required for Java

.NET App Server Agent (includes a Machine Agent by default)	Instrumentation Agent for .NET Common Language Runtime (CLR). This component must be installed on those worker processes that you want to instrument through AppDynamics.	Required for .NET	Required for .NET
PHP Agent	App agent for PHP installations.	Required for PHP	Required for PHP
Machine Agent	Collects hardware performance metrics and can be installed on any machine in your environment. The Machine Agent can be extended to collect data from other subsystems.	Optional	Optional
GeoServer	For End User Management. See <a href="#">Customize Your Web EUM Deployment</a> .	Optional	Optional

## Access to Older Versions

The [AppDynamics Download Center](#) provides downloads of older versions of the products.

- For On-Premise installations: Go to "AD Pro-OnPremise".
- For SaaS installations: Go to "AD Pro-SaaS".

On the top-right corner, click on the drop-down list to select the version that you want to download.

## Downloading from the Linux Shell

To download AppDynamics software from a Linux shell, you can use the wget utility or [cURL](#).

When using these tools, you first need to authenticate to the AppDynamics domain and store the resulting session ID in a file. Next, send the request to download the software, passing the session information file as a cookie.

For example, on Fedora you can use the following wget commands:

```
wget --save-cookies cookies.txt --post-data  
'username=<USERNAME>&password=<PASSWORD>'  
https://login.appdynamics.com/sso/login/
```

```
wget --content-disposition --load-cookies cookies.txt '<URL_TO_FILE>'
```

On the Windows platform add the --no-check-certificate option.

The equivalent cURL commands are:

```
curl -c cookies.txt -d 'username=<USERNAME>&password=<PASSWORD>'  
https://login.appdynamics.com/sso/login/
```

```
curl -O -b cookies.txt <URL_TO_FILE>
```

You can discover the URL for the file to download at the AppDynamics Download Center.

## Learn More

- Supported Environments and Versions
- Agent - Controller Compatibility Matrix

## Quick Start for DevOps

### Get Started

[Get Started on SaaS](#)

[Get Started On-Premise](#)

[Features Overview](#)

### Tutorials for Java

[Use AppDynamics for the First Time with Java](#)

[Understanding Events](#)

[Understanding Flow Maps](#)

[Understanding Slow Transactions](#)

[Understanding the Transaction Scorecard](#)

[Understanding Server Health](#)

[Understanding Exceptions](#)

### Learn More

[Best Practices for Application Developers](#)

[Best Practices for Performance and Quality](#)

Assurance Engineers  
Best Practices for Operations Professionals

## **Monitor Your Applications**

Business Transaction Monitoring  
Web EUM  
Monitor Events  
Monitor Application Change Events  
Background Task Monitoring  
Backend Monitoring  
Infrastructure Monitoring  
Monitor CLRs  
Monitor Hardware

## **Troubleshoot Application Performance**

Troubleshoot Slow Response Times  
Troubleshoot Health Rule Violations  
Transaction Snapshots  
Troubleshoot Node Problems  
Troubleshoot Errors  
Diagnostic Sessions  
Troubleshoot Java Memory Issues

## **Quick Start for Architects**

### **Get Started**

Supported Environments and Versions  
Get Started on SaaS  
Get Started On-Premise

### **Concepts**

Features Overview  
Architecture  
Logical Model  
Mapping Application Services to the  
AppDynamics Model  
Behavior Learning and Anomaly Detection  
Thresholds  
Glossary

### **Basic Configuration**

Configure Business Transaction Detection  
Configure Policies  
Configure Baselines  
Configure Thresholds  
Configure Error Detection

[Set Up and Configure Web EUM](#)  
[Getting Started Wizard for Alerts](#)  
[Custom Dashboards](#)

## Analyze

[Business Metrics](#)  
[Infrastructure Metrics](#)  
[Reports](#)  
[Compare Releases](#)

## Learn More

### Advanced Configuration

[Hierarchical Configuration Model](#)  
[Configure Data Collectors](#)  
[Configure Code Metric Information Points](#)  
[Configure Custom Exit Points](#)  
[Configure Call Graphs](#)  
[Configure Background Tasks](#)  
[Remove Stale Backends](#)  
[Configure Custom Memory Structures for Java](#)  
[Configure JMX Metrics from MBeans](#)  
[Configure Multi-Threaded Transactions \(Java\)](#)  
[Configure Object Instance Tracking for Java](#)  
[Configure Transaction Snapshots](#)  
[Configure Memory Monitoring \(Java\)](#)  
[Internationalization](#)  
[Build an Alerting Extension](#)  
[Export and Import Business Application Configurations](#)

## Integration

[AppDynamics Extensions and Integrations](#)  
[Use the AppDynamics REST API](#)

## Automation

[Workflow Overview](#)  
[Cloud Computing Workflows](#)

## Quick Start for Administrators

### Get Started

[Architecture](#)  
[Get Started on SaaS](#)  
[Get Started On-Premise](#)  
[Logical Model](#)

## Basic Administration

[Release Notes for AppDynamics Pro](#)  
[Supported Environments and Versions](#)  
[Install and Upgrade AppDynamics](#)  
[Name Business Applications, Tiers, and Nodes](#)

### Learn More

## Advanced Administration

[Implement SSL](#)  
[Best Practices for Failover Scenarios for Java](#)  
[Administer Agents](#)  
[Administer the Controller](#)  
[User Authentication and Permissions](#)  
[AppDynamics for Large Enterprises](#)

## Quick Start for Operators

### Get Started

[AppDynamics Essentials](#)

### Tutorials for Java

[Use AppDynamics for the First Time with Java](#)  
[Understanding Events](#)  
[Understanding Flow Maps](#)  
[Understanding Slow Transactions](#)  
[Understanding the Transaction Scorecard](#)  
[Understanding Server Health](#)  
[Understanding Exceptions](#)

### Learn More

[Best Practices for Operations Professionals](#)  
[Set User Preferences](#)  
[Custom Dashboards](#)

## Set User Preferences

- [Change Account Settings](#)
  - To change your password
  - To change your display name and contact email
- [Configure View Preferences](#)
  - To configure view preferences
- [Advanced Features](#)
- [About Debug Mode](#)

Users in the Controller UI can change their passwords, account settings, date and time format, and other user-specific settings in the User Preferences tab, as described by this topic.

## Change Account Settings

The account settings for a Controller UI user include the user's password, display name, and contact email.

Passwords and account settings are attributes of local user accounts (that is, AppDynamics users). If your Controller is configured to use an external authentication mechanism to control access, such as SAML or LDAP, you need to change the equivalent settings in the external system instead.

#### To change your password

1. From the upper right menu bar of the Controller UI, click the **User** icon and then **My Preferences**.
2. Click the **Change Password** button.
3. Enter your current password in the **Current Password** field.
4. Type your new password in the **New Password** and **Confirm New Password** fields, and then click **Save**.

You will need to enter the new password the next time you log in.

#### To change your display name and contact email

The display name is the name that the Controller uses to identify you in certain screen text and messages. For example, it appears in notifications to other Controller users when you share a dashboard with them.

1. In the Controller UI, access your user preferences by clicking the **User** icon and then **My Preferences**.
2. Click the **Edit Account** button.  
Note that your username cannot be changed. To effect a change of a username, you would need to have an administrator delete your account and create another one with the new name.
3. Enter new values for:
  - **Display Name:** Your new display name in the UI.
  - **Email:** The email address where you want to receive notifications from the Controller.
4. Enter your current password in the **Current Password** field. The Controller uses this field to ensure your identity before making changes to your account. If you do not provide the correct password, your changes will not be applied.
5. Click the **Save** button.

The change take effect immediately.

## Configure View Preferences

The Controller UI allows individual users to customize certain view preferences in the UI, such as the time and date format and style elements of the UI.

#### To configure view preferences

1. In the Controller UI, access your user preferences by clicking the **User** icon and then **My Preferences**.
2. In the View Preferences of the page, configure any of the following settings as desired:
  - **Date Format:** By default, the format is MM/DD/YY (for example, 09/25/14). Choose an

- alternate format from the drop-down menu.
- **Use 24 hour Time Format:** Enable this option if you want the UI to represent time in 24-hour time format instead of 12 hour clock format.
  - **Enable Help Pop-ups:** Help popups provide help text in context in the Controller UI. By default, they are enabled. To prevent help popups from appearing in the UI, clear this checkbox. Alternatively, you can prevent individual popups by selecting the **Don't Show Again** checkbox when the popup appears. To clear the list of popups marked as "Don't Show Again", click the **Reset All** button.
  - **Graph Color Scheme for the Metric Browser:** Select either Light or Dark to change the metric browser color scheme.
  - **Graph Color Scheme for All Other Graphs:** Select either **Light** or **Dark** to change the navigation panel color scheme.
  - **Maximum number of Backends to display in graphical views:** This setting limits the number of backend systems that appear in flowcharts or other graphical depictions of your application environment. The default is 20.
  - **Font:** Determines the font type used in the UI. For screen text, the Controller UI uses a font set it embeds and manages by default. If the operating system of the computer on which you access the Controller UI uses a non-English language, you can configure the UI to use non-English languages by setting the font to use system fonts instead. For more information, see [Internationalization](#).
  - **Mouse Wheel Legacy Mode:** If scrolling in the Controller UI using your mouse scroll doesn't work properly, you should try enabling the **Mouse Wheel Legacy Mode** option. This may be necessary if accessing the Controller UI with certain older browsers.
3. You may need to log out of the UI and log back in to see the effects of your changes.

## Advanced Features

AppDynamics cloud automation features allow you to set up workflows that are triggered by policy conditions. By default, the features are hidden in the UI. You need to specifically enable the features to configure cloud auto-scaling features.

To enable cloud automation features in the UI, enable the **Show Cloud Auto-Scaling** option. Enabling this option displays the **Cloud Auto-Scaling** link at the bottom left side of the UI, under the Alert & Respond menu.

See [Workflow Overview](#) for information about using cloud scaling automation features. See [Policies](#) for information about specifying policy conditions that trigger workflows.

## About Debug Mode

The debug mode in the Controller UI is primarily intended for internal use by the AppDynamics development team.

In some cases, you may be asked to enable debug mode in consultation with AppDynamics Support, for example, when you are troubleshooting an issue. However, it is important to note that certain debug mode options can negatively impact Controller performance. For this reason, you should only enable debug mode when directly advised to do so by AppDynamics Support.

## Glossary

- Action
- Agent
- Alert
- Alert Digest
- Anomaly Detection
- Ajax Request
- Application Performance Management
- APM
- App Server
- App Server Agent
- Application
- Application Dashboard
- ART
- Average Response Time
- Backend
- Background Task
- Baseline
- Baseline Deviation
- Baseline Pattern
- BCI
- Browser Snapshot
- Business Application
- Business Metric
- Business Transaction
- Bytecode Injection
- Call Graph
- Compute Cloud
- Controller
- Detection
- Diagnostic Session
- Discovery
- Distributed Application
- End User Monitoring
- End User Response Time
- Entry Point
- Error
- Error Transaction
- Exception
- EUM
- Event
- Exit Point
- Flow Map
- Health
- Health Rule
- Health Rule Violation
- High Availability (HA) Cluster
- Histogram
- Home Page
- iframe

- Information Point
- Key Performance Indicator
- KPI
- Machine
- Machine Agent
- Managed Application
- Match Condition
- Node
- On-Premise
- Pageview
- Policy
- Real-time Business Metric
- Remote Service
- REST
- Request
- SaaS
- Scorecard
- Task
- Tier
- Tag, trace, and learn
- Threshold
- Trace, tracing
- Transaction Correlation
- Transaction Snapshot
- Transaction Splitting
- Workflow

#### Action

An action is an automatic response to an event, based on a policy. There are various types of actions including sending alerts, taking diagnostic snapshots, remediation through scripts, cloud auto-scaling, or custom.

#### Agent

In AppDynamics an agent collects data about an application (and optionally machine) performance or about web page performance. AppDynamics has application server agents (app agents), machine agents, mobile agents, and a JavaScript agent. For example, the App Agent for Java intercepts the bytecode loading at the classloader and enhances it before it is loaded in the JVM. See [Bytecode Injection](#).

#### Alert

An alert notifies a recipient list of a problem or event; by email, SMS or customized to interface with external notification systems.

#### Alert Digest

An alert digest compiles alerts and sends the compilation sent by email or SMS to a recipient list at a configured time interval.

#### Anomaly Detection

Anomaly detection refers to the identification of metrics whose values are out of the normal range, where normal range is based on dynamic baselines that AppDynamics has created based on the previous performance of these metrics.

#### Ajax Request

An Ajax request is a request for data sent from a page using the XHR API. This API is used to send HTTP or HTTPS requests to a web server and to load the server response data back into the requesting page. The Ajax request is tracked as a child of the requesting page using [EUM](#).

#### Application Performance Management

Application performance management monitors and manages the performance and availability of software applications in a production environment, focusing on relating IT metrics to business values.

According to Gartner research, application performance management includes: end user experience monitoring, application runtime architecture discovery and modeling, business transaction management, application component deep-dive monitoring, and application data analytics.

#### APM

See [Application Performance Management](#).

#### App Server

An app server or application server provides software applications with services such as security, data services, transaction support, load balancing, and management of large distributed systems. Examples are a Java Virtual machine (JVM) or a Common Language Runtime (CLR).

#### App Server Agent

An app server agent or app agent monitors an application server. The App Agent for Java monitors a JVM. The App Agent for .NET monitors a CLR. See [Node](#).

#### Application

See [Business Application](#).

#### Application Dashboard

The application dashboard graphically represents high-level structural and status information for a single business application. The application dashboard includes a flow map, grid view, summary of key performance indicators.

#### ART

See [Average Response Time](#).

#### Average Response Time

Average interval between the time the user request is received by the application server and the time that the response is returned to the application server. Does not include the network time for the request to reach the server or the time for the response bytes to reach the caller. Different from

## End User Response Time.

### Backend

A backend or backend node is a software component that is not instrumented directly, but the flow of traffic to it can be monitored. Typical examples are a database or messaging service.

### Background Task

A background task or a batch job is a scheduled program that runs without user intervention.

### Baseline

A baseline provides a defined known point of reference for application performance. The baseline is established by configuration or by observing current performance. Baselines can be static or dynamic.

### Baseline Deviation

A baseline deviation is the standard deviation from a baseline at a point in time. It is represented as an integer value. Baseline deviation can be used to configure health rule conditions based on the number of deviations. For example, you can configure a warning condition as 2 standard deviations from the baseline and a critical condition as 4 standard deviations from the baseline.

### Baseline Pattern

A baseline pattern defines the base time period of data used to create baselines. It can be a fixed time range or rolling time range, in which the most recent x number of days is always used.

### BCI

See [Bytecode Injection](#).

### Browser Snapshot

A browser snapshot presents a set of diagnostic data for an individual base page, iFrame or Ajax request. The data reports the end-user's experience starting with the Web browser. Browser snapshots are taken at periodic intervals and when certain performance thresholds are reached.

### Business Application

An AppDynamics business application models all components or modules in an application environment that provide a complete set of functionality. A business application usually does not map directly to only one Java or .NET or PHP application, and often it maps to more than one.

### Business Metric

See [Real-time Business Metric](#) and [Information Point](#).

### Business Transaction

A business transaction represents an aggregation of similar user requests to accomplish a logical user activity. Examples of these activities include: logging in, searching for items, adding items to

the cart, checking out (e-commerce); content sections that users navigate such as sports, world news, entertainment (content portal); viewing a quote, buying and selling stocks, placing a watch (brokerage). A single request is a business transaction instance.

#### **Bytecode Injection**

Bytecode injection modifies a compiled class at runtime by injecting code into it immediately before it is loaded and run.

#### **Call Graph**

A call graph represents the calling relationships among subroutines in an application. It makes up a part of a transaction snapshot that is used to identify root cause of a performance problem.

#### **Compute Cloud**

A compute cloud delivers computing and storage capacity as a service. Examples are Amazon EC2, OpenStack, etc.

#### **Controller**

The Controller collects, stores, baselines, and analyzes performance data collected by app agents. A Controller can be installed [On-Premise](#) or you can use the AppDynamics [SaaS](#) model.

#### **Detection**

Detection is the process by which AppDynamics identifies a business transaction or backend in a managed application. Detection is also referred to as discovery.

#### **Diagnostic Session**

A diagnostic session is a session in which transaction snapshots are captured, with full call graphs. A diagnostic session can be started manually through the user interface or configured to start automatically when thresholds for slow, stalled, or error transactions are reached.

#### **Discovery**

See [Detection](#).

#### **Distributed Application**

A distributed application runs on multiple computers in a network. Some of the computers can be virtual machines.

#### **End User Monitoring**

End User Experience Monitoring (EUM) provides performance information from the point of view of the client, whether that client is a web browser or a mobile native application. This is different from other types of AppDynamics monitoring, which typically begin at the application server. EUM collects a different set of metrics than the server-side app agents.

#### **End User Response Time**

Average interval between the time that an end-user initiates a request and the completion of the page load of the response in the user's browser. In the context of an Ajax request, the interval

ends when the response has been completely processed. Not to be confused with [Average Response Time](#).

#### Entry Point

An entry point begins or extends a business transaction. An entry point is usually a method or operation in your application code. AppDynamics automatically detects entry points for common frameworks, and you can configure entry points to customize how AppDynamics detects business transactions.

#### Error

An error in AppDynamics indicates an unhandled exception in the context of a business transaction, a logged error of the appropriate severity, or any exception called during an exit call, which prevents the transaction from working properly.

#### Error Transaction

An error transaction is an error that occurred during transaction execution. An error transaction can be caused by a logged error or a thrown exception.

#### Exception

An exception is a code-based anomalous or exceptional event, usually requiring special processing. It can occur in the context of a business transaction and outside of a business transaction

#### EUM

See [End User Monitoring](#).

#### Event

An event represents an action or occurrence detected by the system that can be handled by the system. There are different event types.

#### Exit Point

An exit point is a call from an app server to a backend database, remote service or to another app server. AppDynamics automatically detects many exit points and you can configure custom exit points.

#### Flow Map

A flow map graphically represents the tiers, nodes, and backends and the process flows between them in a managed application.

#### Health

Health in AppDynamics refers to the extent to which the application being monitored operates within the acceptable performance limits defined by health rules. Health is indicated by a green/yellow/red color scheme.

#### Health Rule

Health rules allow you to select specific metrics as key to the overall health of an application and to define ranges for acceptable performance of those metrics. AppDynamics supplies default health rules that you can customize, and you can create new ones.

#### **Health Rule Violation**

A health rule violation exists if the conditions of a health rule are true.

#### **High Availability (HA) Cluster**

A cluster of computers that hosts duplicate server applications with the purpose of reducing down time. The HA cluster, also called a failover cluster, is enabled by redundant systems that guarantee continued delivery of service during system failure.

#### **Histogram**

A histogram is a graphical representation of the distribution of data, shown as adjacent rectangles, erected over discrete intervals (bins), with an area proportional to the frequency of the observations in the interval.

#### **Home Page**

The Web page you see when you first log into the AppDynamics Controller, before you have selected an application. See [AppDynamics Home Page](#).

#### **iframe**

An iframe is an "inline frame", an HTML document that is embedded in another HTML document. It is tracked as a child page using EUM.

#### **Information Point**

An information point instruments a method in application code outside the context of any business transaction. It is used for monitoring the performance of the method itself or for capturing data from the method's input parameters or return value.

#### **Key Performance Indicator**

Key performance indicators are main metrics that an organization uses to measure its success. In AppDynamics, the key performance indicators are assumed to be load (number of calls and calls per minute), average response time, and errors (number of errors and errors per minute.)

#### **KPI**

See [Key Performance Indicator](#).

#### **Machine**

A machine consists of hardware and an operating system. It hosts application services and it can be virtual.

#### **Machine Agent**

A machine agent instruments a machine to report data about hardware and the network to the Controller. AppDynamics provides both a Standalone Machine Agent and an embedded machine

agent in the App Agent for .NET. The Standalone Machine Agent functionality can be extended to add additional metrics.

#### Managed Application

A managed application is an application with servers that are instrumented by AppDynamics.

#### Match Condition

A match condition frames a test consisting of a match criterion (such as a method name, servlet name, URI, parameter, hostname, etc.), a comparison operator typically selected from a drop-down list, and a value. Used in many types of AppDynamics configuration to specify entities to be included in or excluded from monitoring.

#### Node

A node is an instrumented Java application server or an instrumented Windows .NET application (IIS, executable, or service.) Instrumentation is accomplished by installing an AppDynamics App Agent. Nodes belong to tiers. See [Tier](#).

#### On-Premise

On-Premise refers to an AppDynamics Pro installation where the controller is installed on machines at your site. Alternatively, AppDynamics offers AppDynamics Pro as an SaaS.

#### Pageview

A pageview is an instance of a web page being loaded into a Web browser.

#### Policy

A policy consists of a trigger based on events and an action respond to the event. A policy provides a mechanism for automating monitoring, alerting, and problem remediation.

#### Real-time Business Metric

Metric that measures items such as revenue per transaction, number of orders, number of credit card purchases and so on. Differ from a performance metric, which measure the performance of the application. Implemented through [Information Pointss](#).

#### Remote Service

A remote service provides a service to a distributed application outside of the JVM or CLR. Examples are a Java Message Service or Web Service. See [Backend](#).

#### REST

The AppDynamics REST API is implemented using Representational State Transfer (REST) Services. You use the REST API to retrieve information from AppDynamics programmatically.

#### Request

A request is a single instance of a business transaction; for example, 500 requests per minute for the "Checkout" business transaction.

#### SaaS

SaaS is an acronym for Software as a Service. AppDynamics provides AppDynamics Pro as an SaaS where the controller is hosted on AppDynamics in-house machines and monitors your applications, communicating over the internet with app server, Java, .NET, infrastructure, and database agents installed in your environments. You can, alternatively, install the AppDynamics Pro controller on your own equipment, an installation type referred to as [On-Premise](#).

#### Scorecard

A visual summary of the performance of a business transaction within a specified time range, covering percentage of instances that are normal slow, very slow, stalled or errors.

#### Task

A task codifies a unit of work as a set of instructions and is a component of a step in a workflow.

#### Tier

A tier represents a key service in an application environment, such as a website or processing application. A tier is composed of one or more nodes or one or more backends. See [Node](#) and [Backend](#). An "originating tier" is the tier that receives the first request of a business transaction. A "downstream tier" is a tier that is called from another tier.

#### Tag, trace, and learn

Tag, trace, and learn is a methodology used for tracing code execution and discovering the business transaction context.

#### Threshold

A threshold is a configurable boundary of acceptable or normal business transaction or background task performance.

#### Trace, tracing

Tracing is following the execution of software code and recording information about the execution, usually for debugging or performance monitoring purposes.

#### Transaction Correlation

Transaction correlation is the internal mechanism that allows AppDynamics to do transaction tracing in modern web applications, tying together distributed components into a single entity, the business transaction, for monitoring purposes.

#### Transaction Snapshot

A transaction snapshot depicts a set of diagnostic data for an individual business transaction across all app servers through which the business transaction has passed. The data reports the user's experience starting with the application server. A transaction snapshot is taken at a specific point in time.

#### Transaction Splitting

The process of using a dynamic value to customize how Business Transactions are identified.

#### Workflow

A workflow builds a sequence of steps in which each step consists of one or more tasks that are executed on a machine instrumented by a Standalone Machine Agent.

## Tutorials for Java

This section provides tutorials for tasks in AppDynamics.

### Quick Tour of the User Interface



### Troubleshooting Application Errors

Identifying and troubleshooting errors in your Java application.

## Overview Tutorials for Java

[Quick Tour of the User Interface](#)



## Use AppDynamics for the First Time with Java

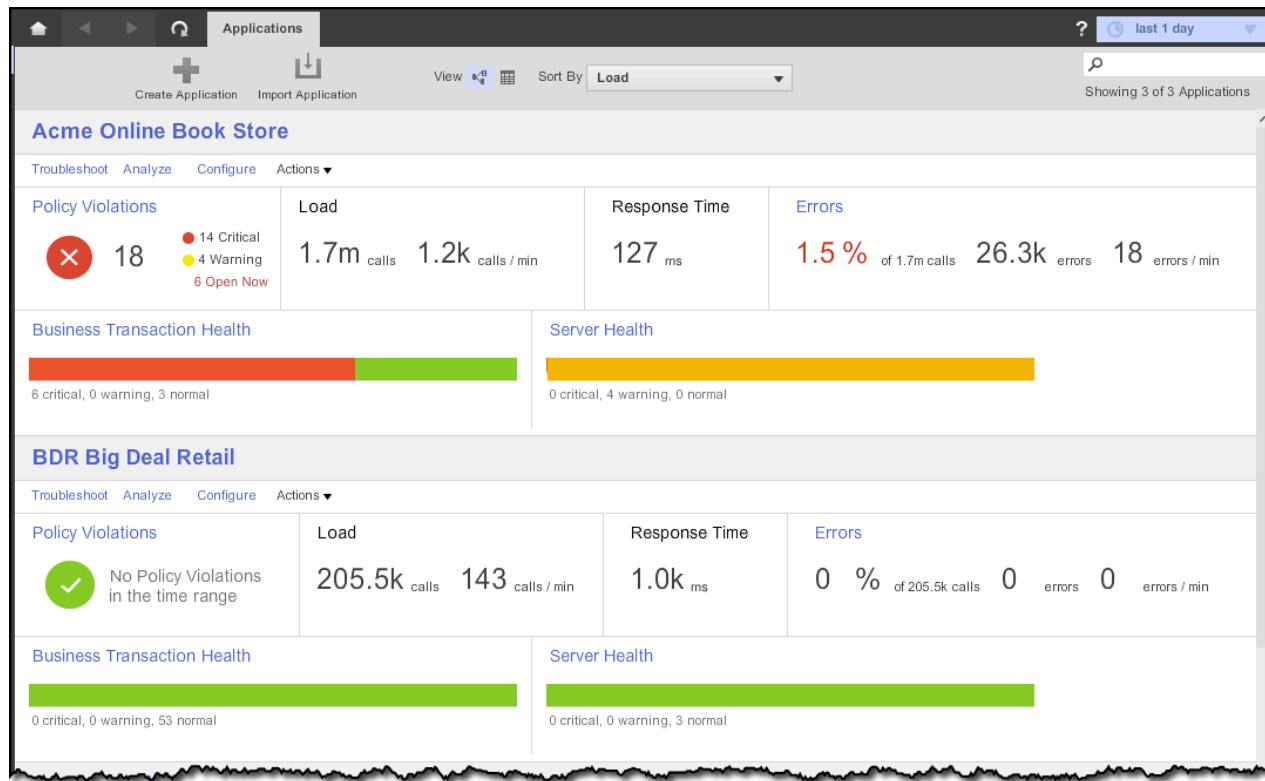
- [All Applications Dashboard](#)
- [Application Dashboard](#)
  - Time Range
  - Flow Map and KPIs
  - Events
  - Transaction Scorecard
  - Exceptions and Errors
- [More Tutorials](#)

This topic assumes that an application is already configured in AppDynamics, and uses the Acme Online application as the example. It also assumes that you have already logged in to AppDynamics.

This topic gives you an overview of how AppDynamics detects actual and potential problems that users may experience in your application - transactions that are slow, stalled or have errors - and helps you easily identify the root causes.

### All Applications Dashboard

When you log into the Controller UI you see the All Applications dashboard.



The All Applications dashboard shows high-level performance information about one or more business applications. Load, response time, and errors are standard metrics that AppDynamics calls "key performance indicators" or "KPIs". The others are:

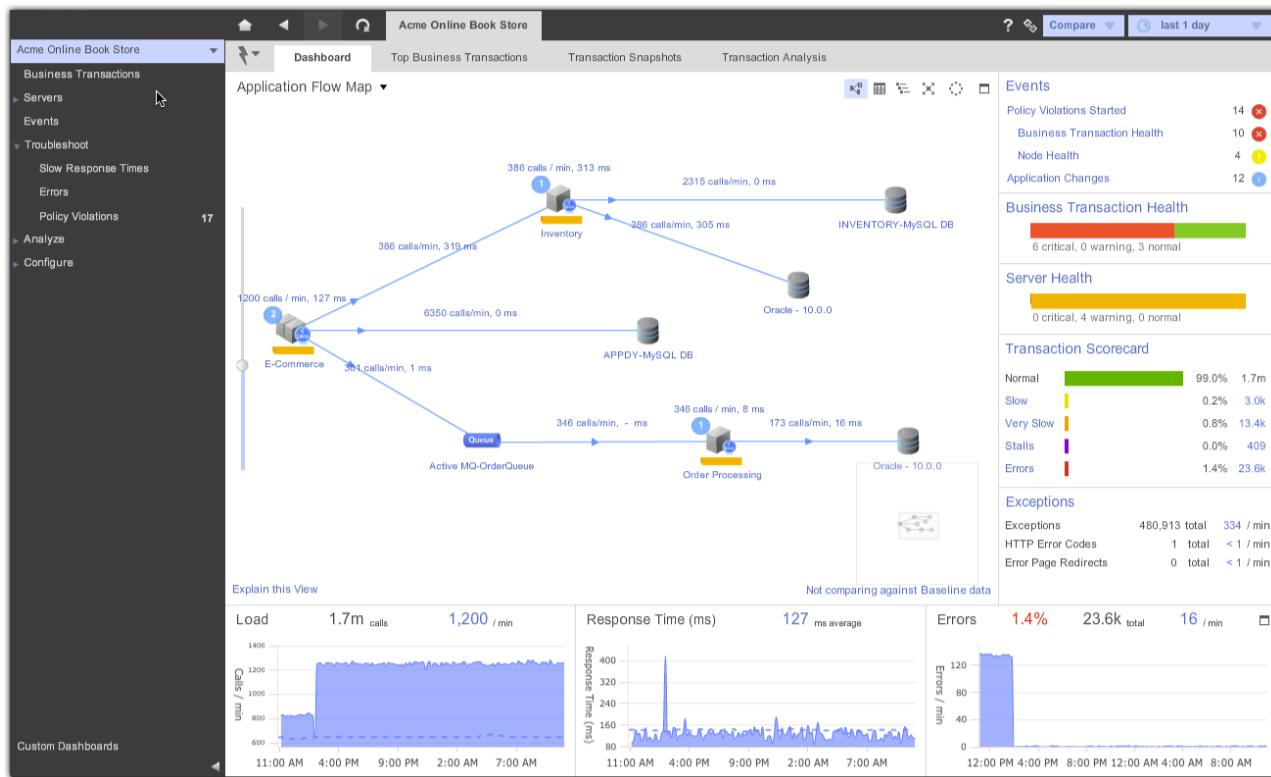
**Health Rule Violations and Policies:** AppDynamics lets you [define a health rule](#), which consists of a condition or a set of conditions based on metrics exceeding predefined thresholds or dynamic baselines. You can then use health rules in policies to automate optional remedial actions to take if the conditions trigger. AppDynamics also provides default health rules to help you get started.

**Business Transaction Health:** The health indicators are a visual summary of the extent to which a business transaction is experiencing critical and warning health rule violations. See the [slow transactions tutorial](#).

**Server Health:** Additional visual indicators that track how well the server infrastructure is performing. See the [server health tutorial](#).

## Application Dashboard

Click an application to monitor, one that has some traffic running through it. The Application dashboard gives you a view of how well the application is performing.



You see the dashboard for your application. The flow map on the left gives you an overview of your servers (application servers, databases, remote servers such as message queues, etc.) and metrics for the calls between them. Click, hold and move the icons around to arrange the flow map. Use the scale slider and mini-map to change the view.

## Time Range

From the time range drop-down in the upper-right corner select the time range over which to monitor - the last 15 minutes, the last couple of hours, the last couple of days or weeks. Try a few different time ranges and see how the dashboard data changes.

## Flow Map and KPIs

In the flow map, click any of the blue lines to see more detail on the aggregated key performance metrics (load, average response time and errors) between the two servers. See the [flow maps tutorial](#).

The graphs at the bottom of the dashboard show the key performance indicators over the selected time range for the entire application.

## Events

An event represents a change in application state. The Events pane lists the important events occurring in the application environment. See the [events tutorial](#).

## Transaction Scorecard

The Transaction Scorecard panel shows metrics about business transactions within the specified time range, covering the percentage of instances that are normal slow, very slow, stalled or have

errors. Slow and very slow transactions have completed. Stalled transactions never completed or timed out. [Configurable thresholds](#) define the level of performance for the slow, very slow and stalled categories. See the [Transaction Scorecard tutorial](#).

## Exceptions and Errors

An exception is a code-logged message outside the context of a business transaction. An error is a departure from the expected behavior of a business transaction, which prevents the transaction from working properly. See the [exceptions tutorial](#).

## More Tutorials

## Monitoring Tutorials for Java

### Tutorial for Java - Events

- Monitoring Events
- Filtering Events

## Monitoring Events

1. From an application, tier or node dashboard, look at the **Events** panel.

The screenshot shows the AppDynamics interface for monitoring Java applications. On the left, there's a navigation sidebar with links like MyApp, Business Transactions, Servers, and Events. The 'Events' link is circled in green. Below it, under 'Analyze', are links for Troubleshoot, Slow Response Times, Errors, Policy Violations, and Configuration. Under 'Configure', there are links for Policies, Alerts, Slow Transaction Thresholds, and Baselines. At the bottom of the sidebar is a 'Custom Dashboards' section.

The main content area is titled 'Events' and shows a table of 8 events. The columns are Type, Summary, Time, Business Transaction, Tier, and Node. The events listed are:

Type	Summary	Time	Business Transaction	Tier	Node
Code Deadlock	JVM deadlock det...	09/24/12 9:44:36 AM		2Tier	node2
Code Deadlock	JVM deadlock det...	09/24/12 9:39:36 AM		2Tier	node2
Code Deadlock	JVM deadlock det...	09/24/12 9:34:36 AM		2Tier	node2
Code Deadlock	JVM deadlock det...	09/24/12 9:29:36 AM		2Tier	node2
Code Deadlock	JVM deadlock det...	09/24/12 9:24:36 AM		2Tier	node2
Code Deadlock	JVM deadlock det...	09/24/12 9:19:36 AM		2Tier	node2
Slow Requests - Very Slow	/processor/order/elect...	09/24/12 9:17:08 AM	/processor/order/elect...	1Tier	node1
Slow Requests - Very Slow	/product/outdoorM...	09/24/12 9:17:08 AM	/product/outdoor...	1Tier	node1

A large green oval encloses the event list area. Inside the oval, a callout box states: "The Events Panel shows the type of event, a synopsis, the timestamp when the event occurred, what business transaction the event is associated with (if any), the source of the event by tier and node."

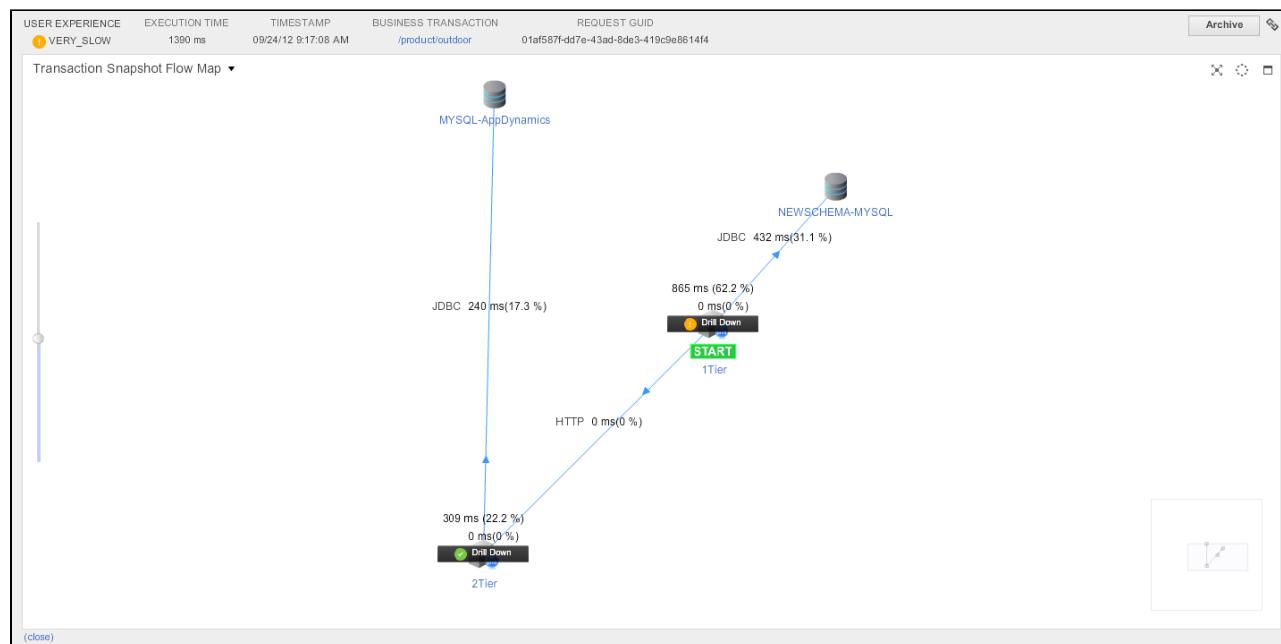
The **Events** panel shows all the events that are monitored by AppDynamics. There are several types of events:

- **Health Rule Violation Events** include business transaction health rule events such as average response time, and server health events such as Java VM Heap Utilization Thresholds. To change what generates a health rule event, see [Health Rules](#).
- **Slow Transaction Events** occur when slow, very slow or stalled transactions are detected. See [Configure Thresholds](#).
- **Error Events** occur when application exceptions are thrown or HTTP Errors are returned.

See [Configure Error Detection](#).

- **Code Problem Events** occur when a code deadlock is detected or a resource pool is utilized beyond the specified threshold. For example this event occurs when a JDBC connection pool is above 80% utilized or when a java.lang.Thread is deadlocked.
- **Application Change Events** occur when administrative changes are made to an application tier. For example, this event occurs when an application server instance is restarted or when a Java VM option is modified on an application server. See [Monitor Application Change Events](#).
- **AppDynamics Config Warnings** occur when the AppDynamics infrastructure needs attention. For example, when the Controller detects low disk space conditions on its host the policy associated with this event type occurs. See [AppDynamics Administration](#).
- **Custom events** are user-defined events. See [Events](#).

2. To get details click an event in the list. For example, click a slow request event to see the details of the request in the transaction flow map so you can start troubleshooting the slow request.



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

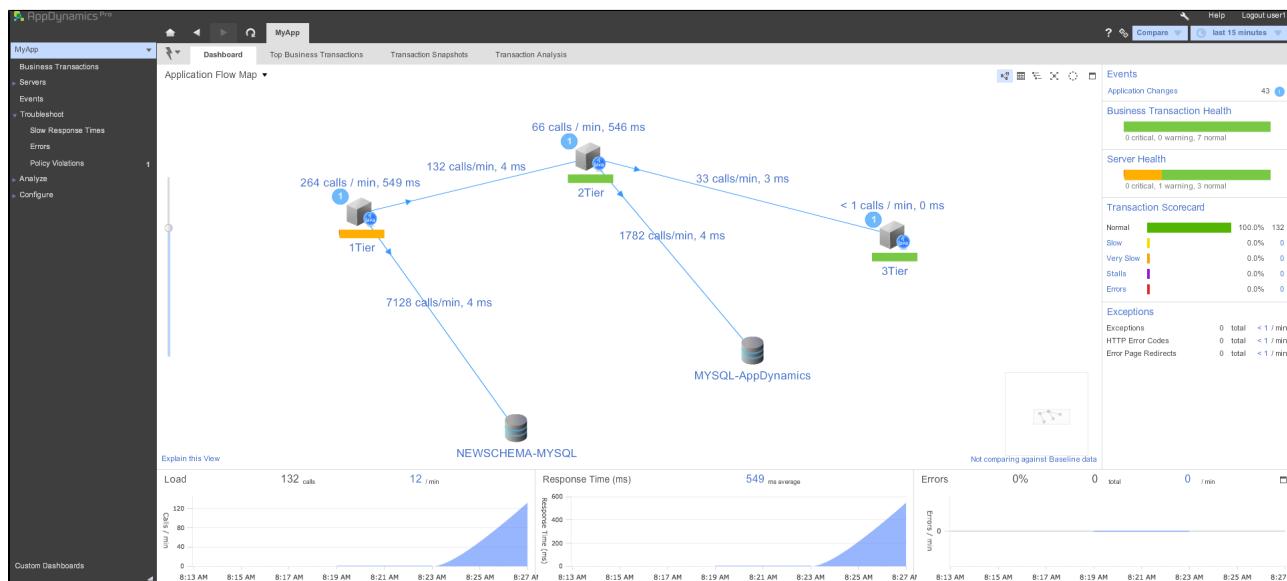
## Filtering Events

You can filter events by type in the **Events** panel. Click the type of event you want to see and click **Search**. For example, to see only **Deadlocks** and **Resource Pool Exhaustion** events select the **Code Problem Event** and click **Search**.

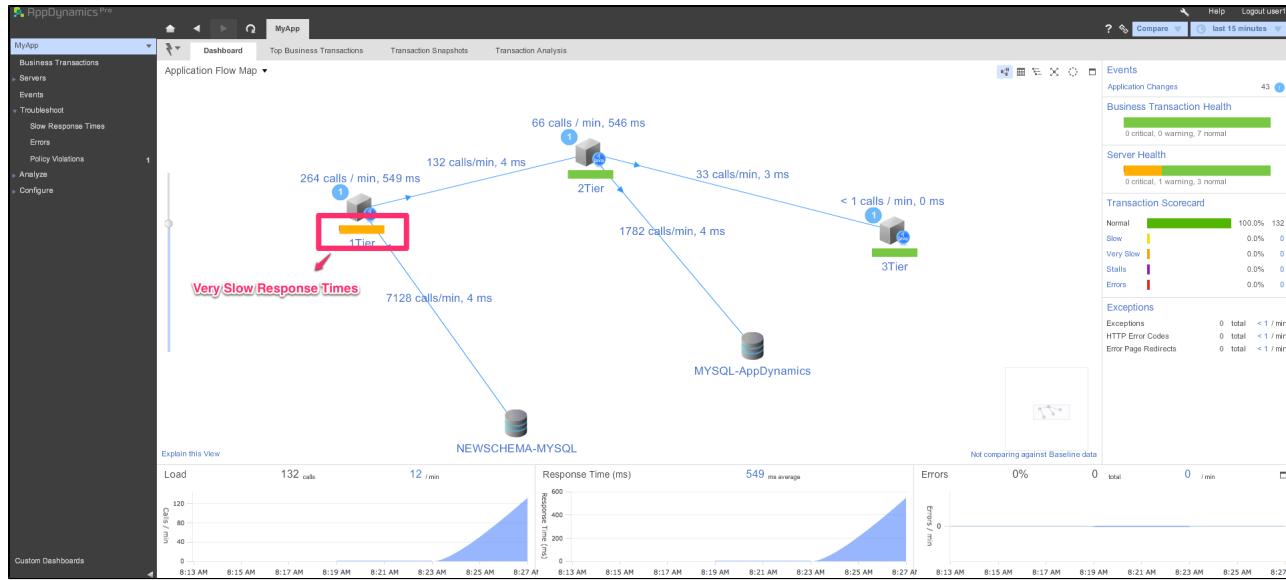
The screenshot shows the AppDynamics Events page for the 'MyApp' application. On the left, the navigation menu includes 'Business Transactions', 'Servers', 'Events', 'Troubleshoot', 'Analyze', and 'Configure'. The 'Events' section is selected. In the center, there's a search bar with 'Search' and 'Clear Criteria' buttons. Below the search bar is a 'FILTER BY EVENT TYPE' section with several checkboxes. The 'Code Problems' checkbox is checked, and its sub-options 'Code Deadlock' and 'Resource Pool Limit Reached' are also checked. A red arrow points to this section. To the right is a table titled 'Register Application Change Event' with columns for Type, Summary, Time, Business Transaction, Tier, and Node. Six events are listed, all of which are 'Code Deadlock' events from 'JVM deadlock detected' on 'node2' in '2Tier'. A green callout bubble points to the table with the text 'The list just shows selected event types'.

## Tutorial for Java - Flow Maps

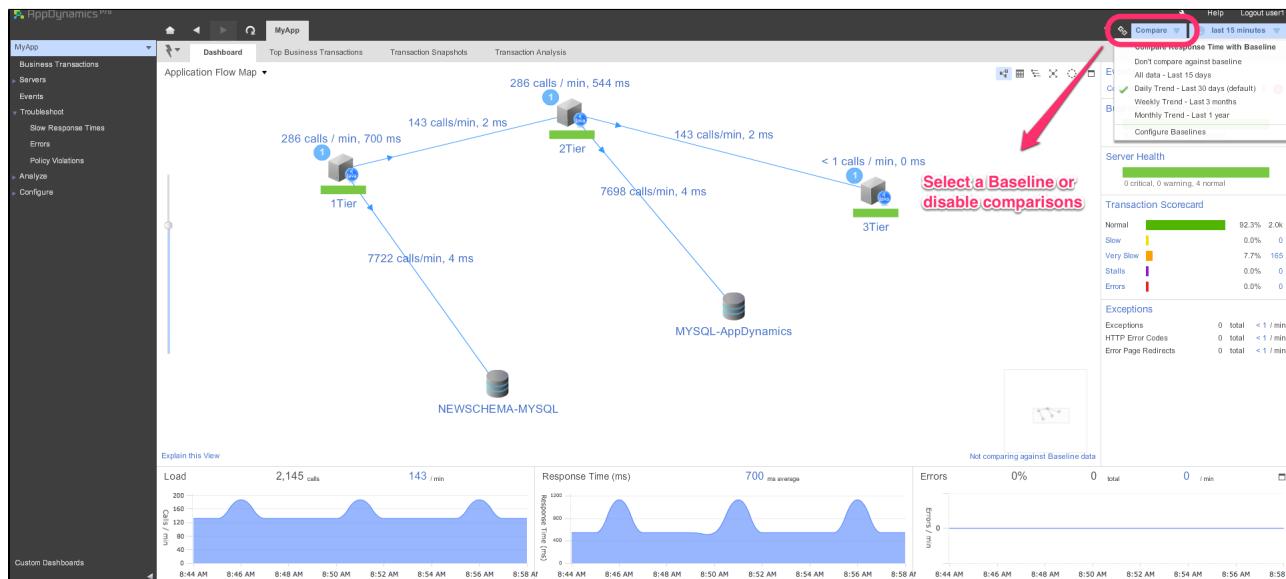
Flow maps show the health of your application, all tiers, and the communication between the tiers. It includes summary indicators such as call rates and error rates:



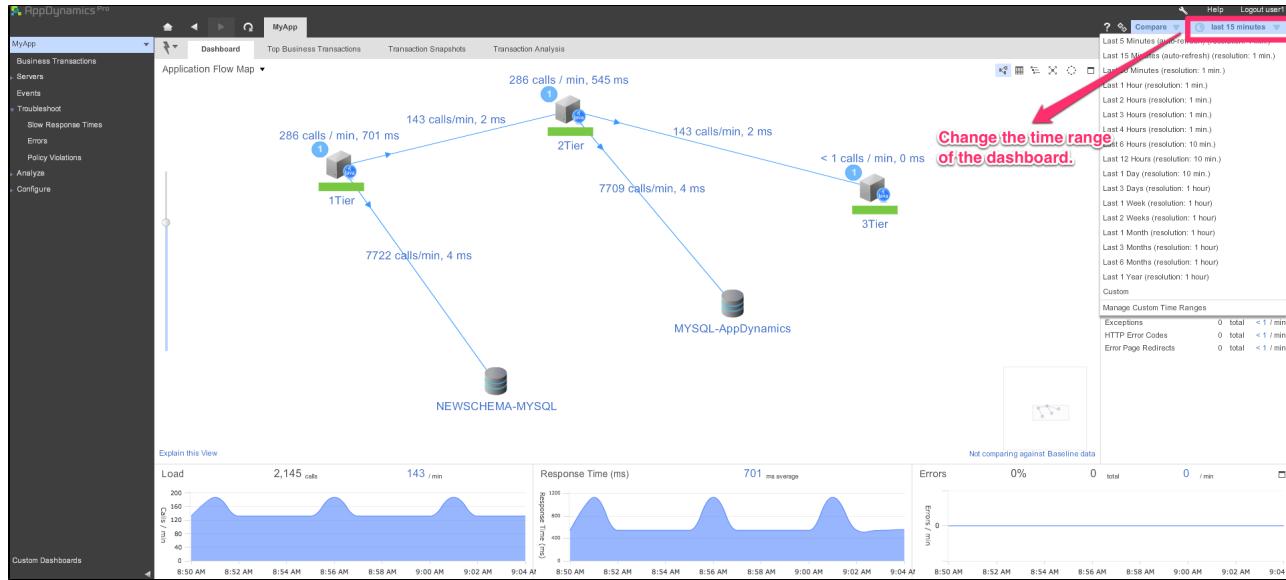
Visual indicators quickly show you problem tiers and healthy tiers. Below you can see tier 1 is experiencing very slow response times, while tier 2 and tier 3 are healthy:



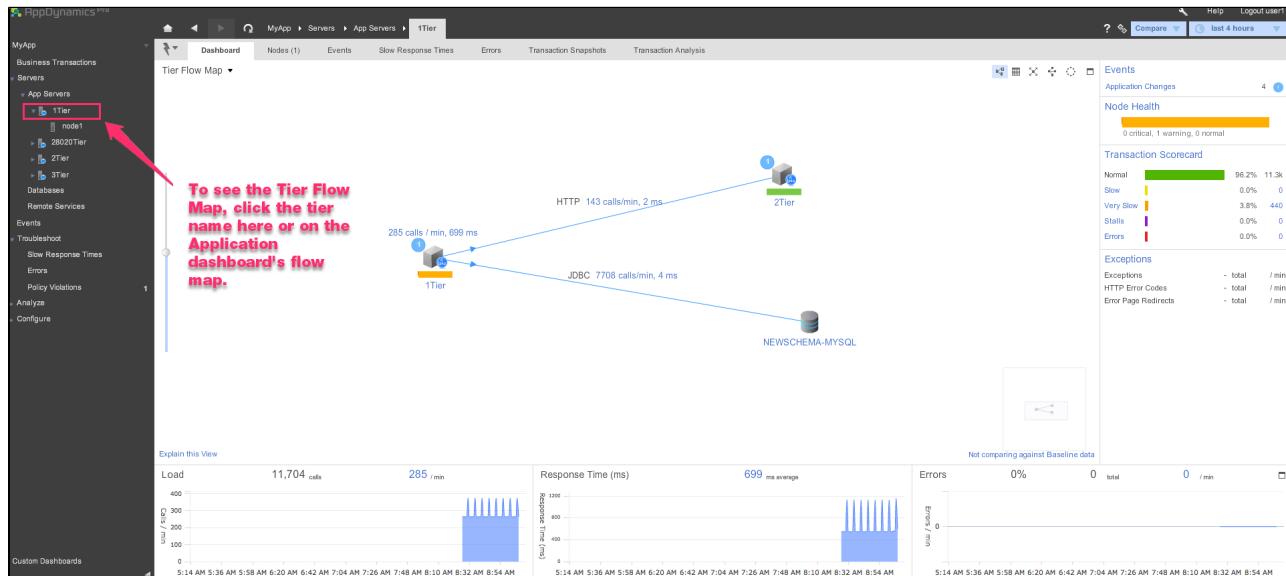
By default, the flow map computes tier health by comparing the state of the tier averaged over the last 15 minutes against the daily trend (the 30 day rolling average). You can change the time window for baseline comparison using the time window pull down menu. You can also disable baseline comparisons:



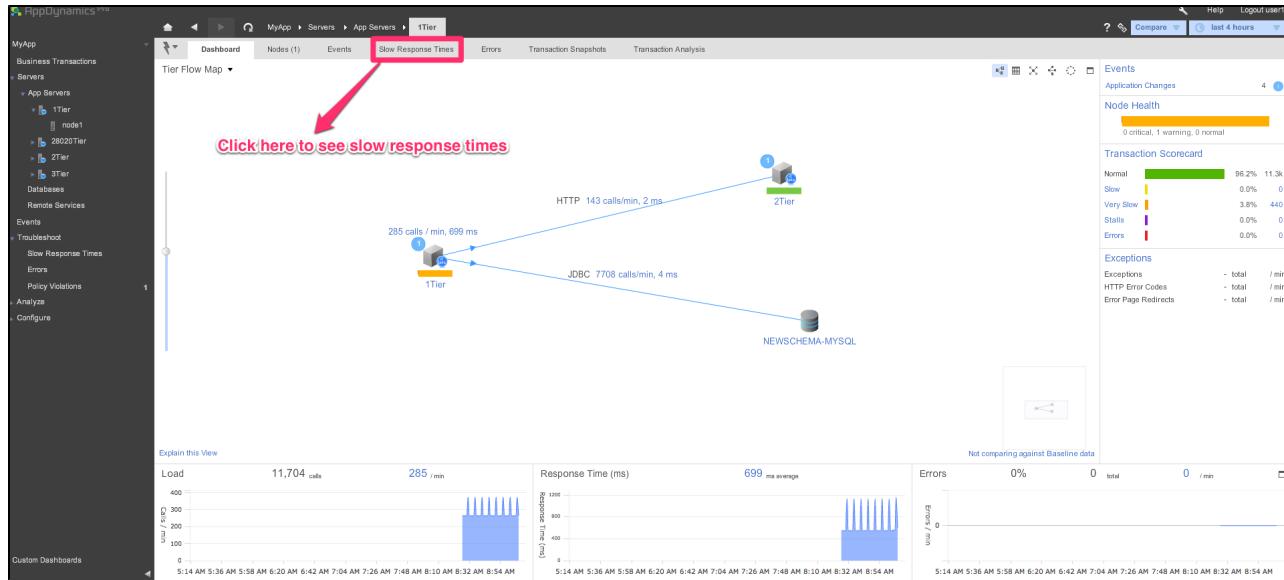
You can change the time range displayed in the flow map by changing the time window using the time window pull down menu. Changing the time range effects the entire dashboard:



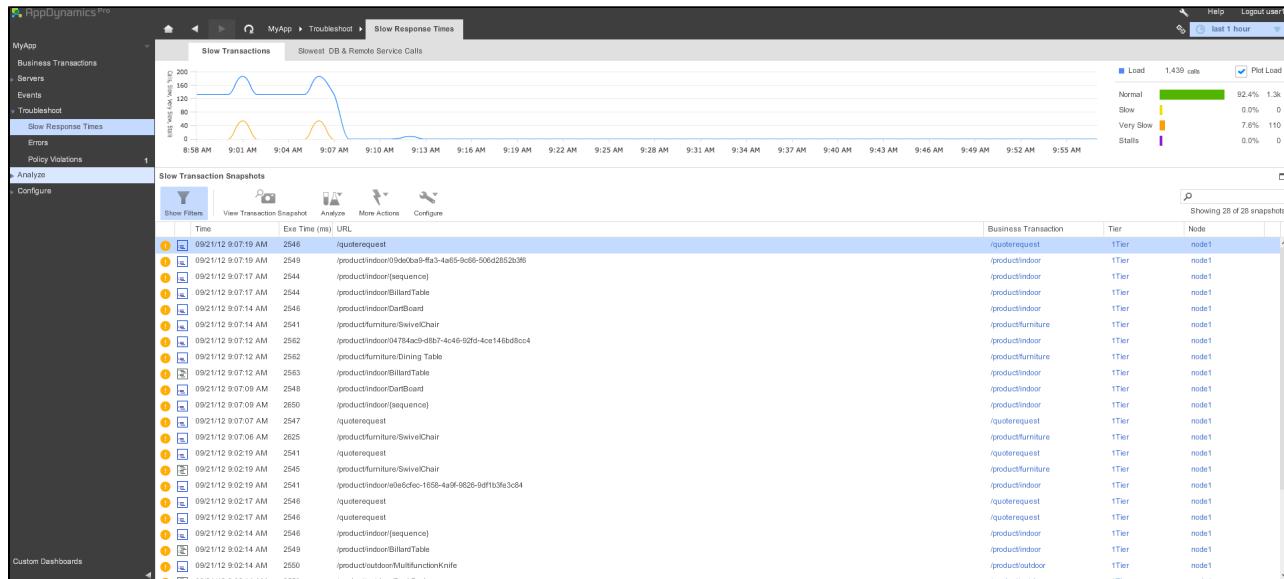
You can troubleshoot a problem system call by clicking on a the tier's name to drill down into a subset of the system involving the tier:



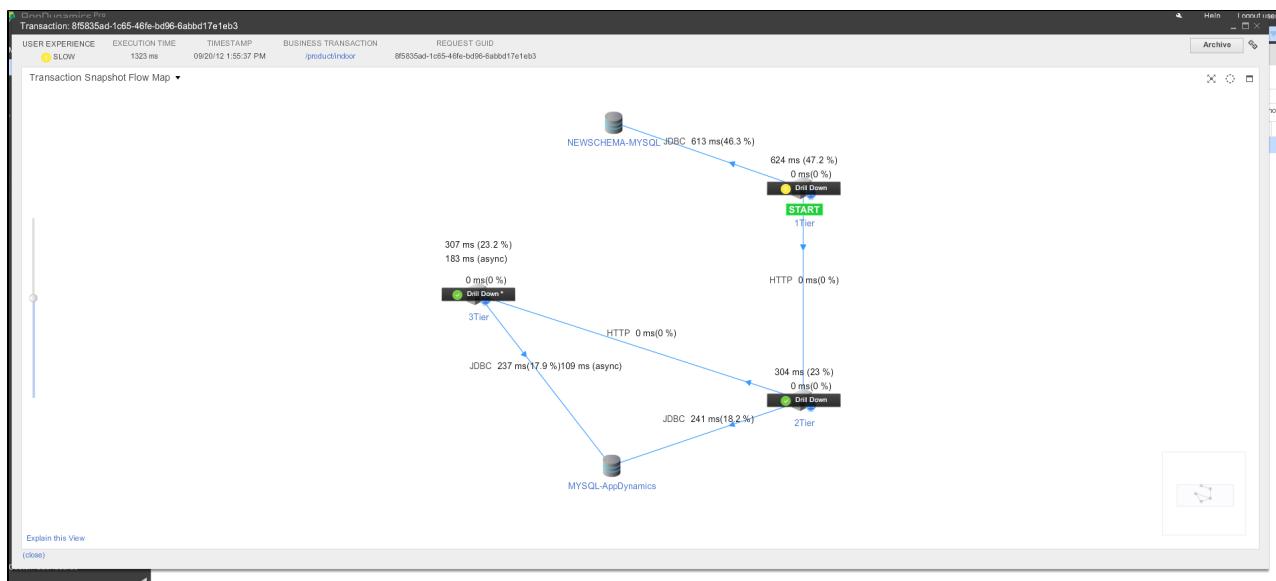
To view the slow response times in detail click on the slow response time menu:



From the slow response time pick a transaction to see a snapshot of the slow transaction:



From the transaction snapshot you can troubleshoot the slow transaction:



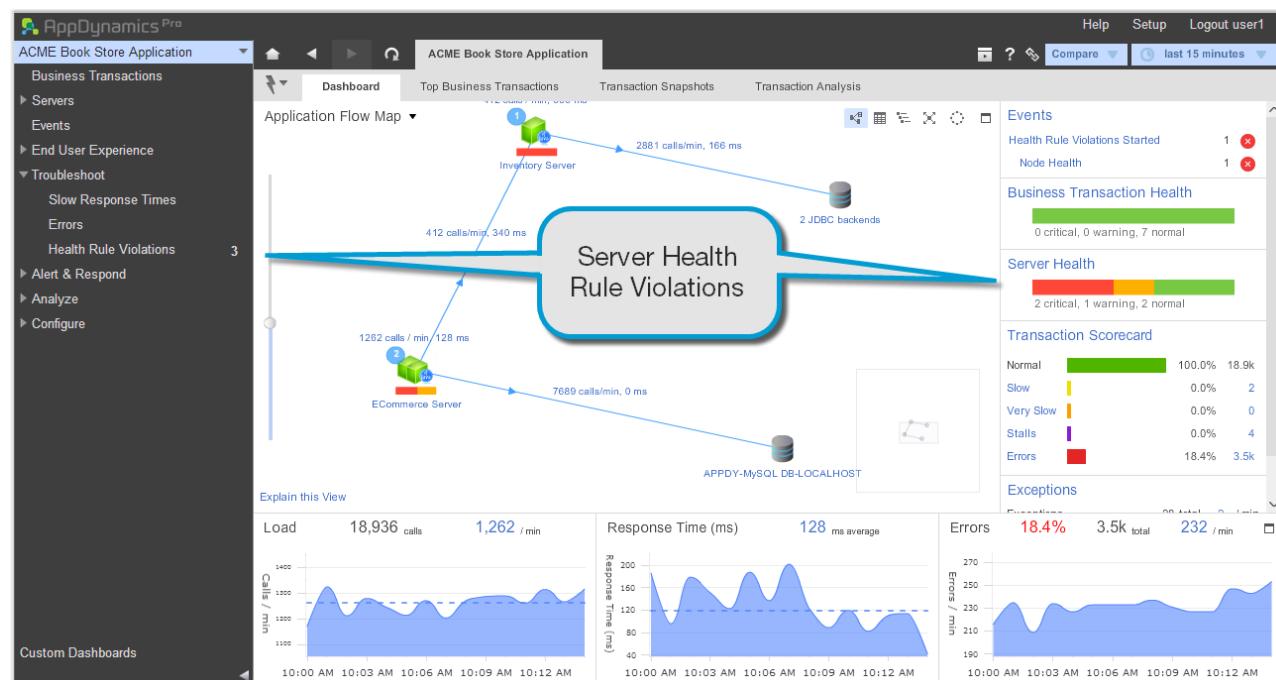
For more information on resolving issues related to slow transactions, see [Troubleshoot Slow Response Time for Java](#).

## Tutorial for Java - Server Health

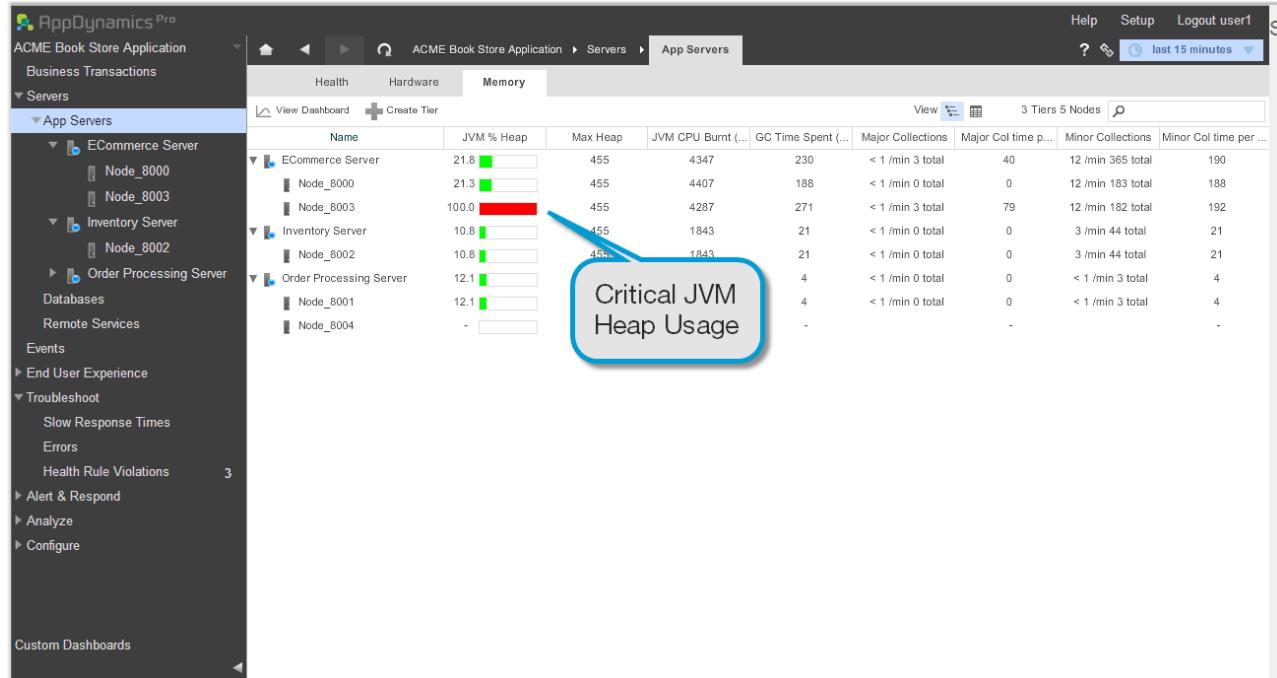
- About Java Server Health
- Learn More

### About Java Server Health

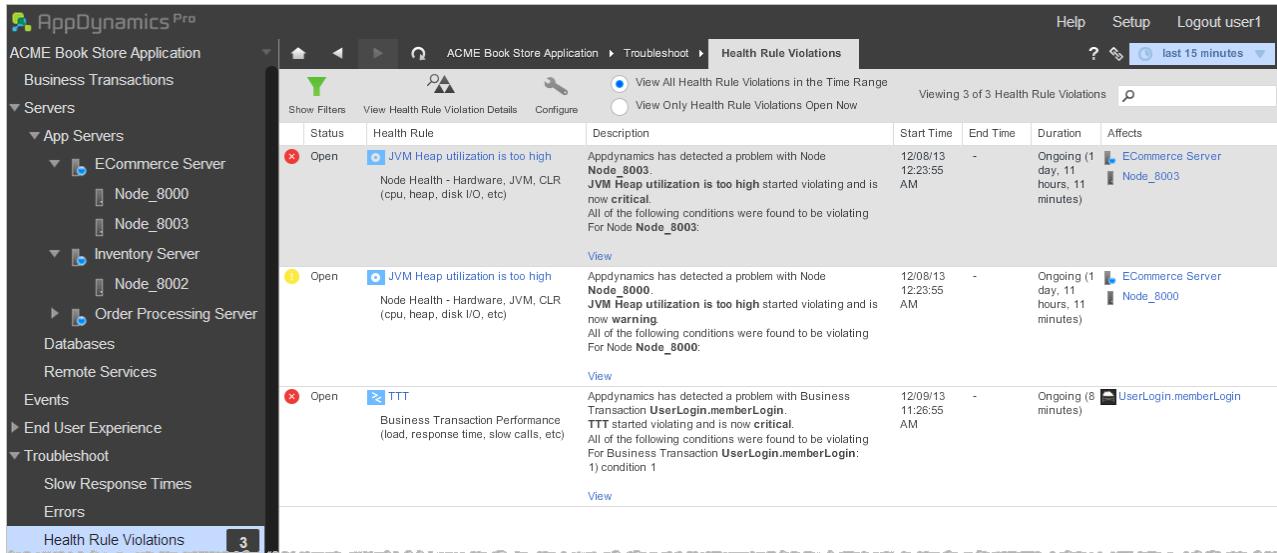
By default, AppDynamics provides predefined rules for CPU utilization, physical memory utilization, JVM heap utilization, and CLR heap utilization. For example the default health rule for CPU utilization triggers a warning when a node exceeds 75% CPU utilization and triggers a critical event when CPU utilization is 90% or above.



Node health is driven by node health rules. The following example shows that Node\_8003 is experiencing a JVM heap health rule violation; all of the heap is consumed.



You can view the health rule violation details and the status of the violation:



There are many types of health rules and each defines a condition or set of conditions in terms of a certain set of metrics that serve as health indicators of your application component. For example, the Business Transaction Performance health rule defines a set of conditions in terms of business transaction metrics, while the Node Health-Hardware,JVM,CLR health rule defines a set of conditions in terms of hardware metrics.

To change or add a new health rule, see [Configure Health Rules](#).

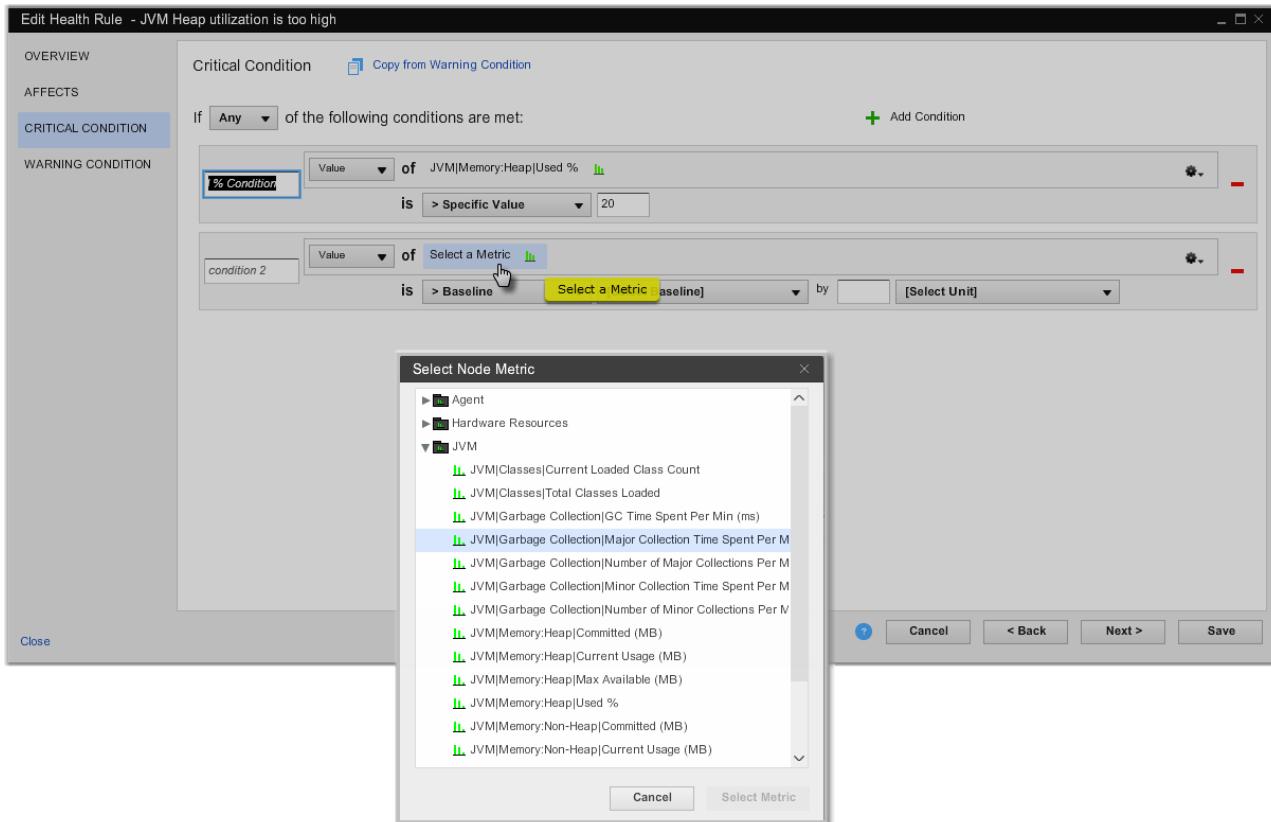
The screenshot shows the 'Edit Health Rule - JVM Heap utilization is too high' dialog. The 'OVERVIEW' tab is selected. The rule is named 'JVM Heap utilization is too high'. It is enabled and affects 'Tiers'. The 'Type' dropdown is set to 'Overall Application Performance (load, response time, num slow calls, etc)'. The 'CRITICAL CONDITION' section includes 'Business Transaction response time is much higher than normal' and 'CPU utilization is too high'. The 'WARNING CONDITION' section includes 'Memory utilization is too high' and 'CLR Garbage Collected high'. The 'When is this Health Rule Enabled?' section shows 'Always' checked. The 'During these times:' dropdown is set to 'End of Business Hour: 5pm-6pm, Mon-Fri'. The 'Use the last' dropdown is set to '30' minutes. The 'Wait Time after Violation' section indicates 'Health Rules are evaluated every minute.' The bottom right of the dialog has buttons for 'Cancel', '< Back', 'Next >', and 'Save'.

You can control the scope of a health rule to a specific application component. For example, for Node Health, you can choose between scoping to tiers or nodes. For tiers, you can apply the health rule to all tiers or to specific tiers only. For nodes, you can apply the health rule to all nodes. If you have a large cluster, you can choose specific nodes as well.

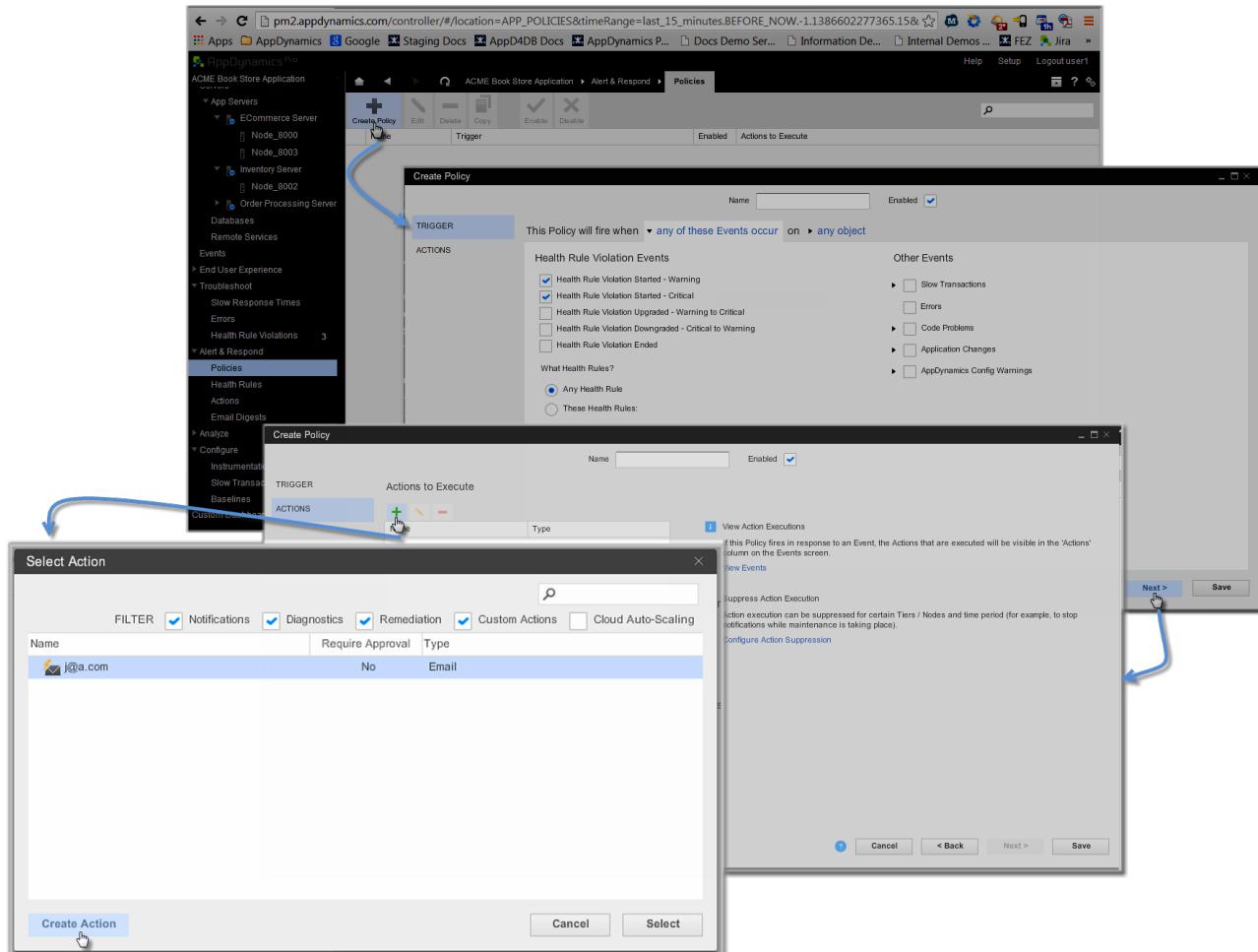
The screenshot shows the 'Edit Health Rule - JVM Heap utilization is too high' dialog with the 'AFFECTS' tab selected. The 'What does this Health Rule affect?' section has the 'Nodes' radio button selected. The 'Select what Nodes this Health Rule affects' section shows 'Type of Nodes' set to 'Java Nodes'. A dropdown menu shows 'All Nodes in the Application' is selected. Below it, a note states 'This Health Rule affects all Nodes in the Application: ACME Book Store Application'. The bottom right of the dialog has buttons for 'Close', 'Cancel', '< Back', 'Next >', and 'Save'.

Once you scope the health rule, you can define the triggering conditions of the health rule. There are two status condition sets, warning and critical, which can be defined independently of each other. Each status condition set consists of atomic conditions that must either be ALL met or have ANY that are met in order to trigger the status condition. Atomic conditions are based on

predefined metrics that serve as health indicators of your application component. If you cannot find a Health Rule type that has a predefined metric that meets your needs, you can [add a metric using custom monitors](#) or [create a JMX metric from MBeans](#) and use a Custom Health Rule.



After defining Health Rules, you can use the Health Rule Violation Events in policies to trigger [actions](#) that can notify Administrators via email or SMS systems or perform some other action.



## Learn More

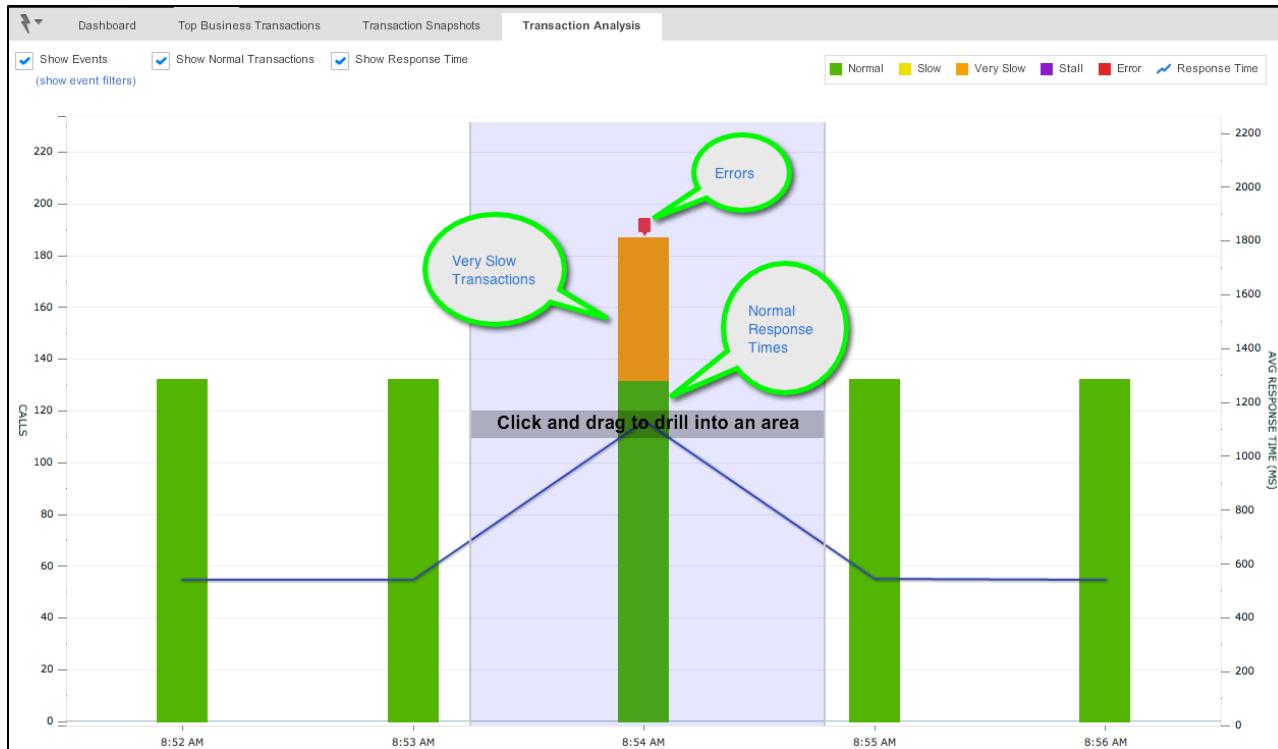
- Troubleshooting Server Health, AppDynamics in Action video
- Tutorial for Java - Transaction Scorecards**

The screenshot shows the AppDynamics Pro interface for the application 'MyApp'. The left sidebar includes sections for Business Transactions, Servers, Events, Troubleshoot (Slow Response Times, Errors, Policy Violations), Analyze, and Configure. The main area features an 'Application Flow Map' illustrating a two-tier architecture. The flow starts at a '1Tier' node, goes to a '2Tier' node, and then connects to a 'MYSQL-AppDynamics' database. Call volumes and response times are displayed along each segment: 286 calls/min, 698 ms between 1Tier and 2Tier; 143 calls/min, 2 ms between 2Tier and MySQL; and 286 calls/min, 545 ms from MySQL back to 2Tier. A green callout bubble highlights the 'Business Transaction Scorecard' section on the right. This section contains a table showing transaction performance metrics:

Category	Normal	Slow	Very Slow	Stalls	Errors
Normal	92.3%	660			
Slow	0.0%	0			
Very Slow	7.7%	55			
Stalls	0.0%	0			
Errors	0.0%	0			

Below the scorecard are sections for 'Exceptions' (0 total exceptions), 'HTTP Error Codes' (0 total errors), and 'Error Page Redirects' (0 total errors). At the bottom, three time-series charts show 'Load' (Calls/min), 'Response Time (ms)' (ms average), and 'Errors' (Errors/min) over the period 8:45 AM to 8:49 AM.

Transactions are categorized as Normal, Slow, Very Slow, Stalled, or Errors, which are determined by thresholds and the AppDynamics error detection subsystem. Thresholds can be static or dynamic; dynamic thresholds are based on historical data. The Transaction Analysis Histogram shows the distribution over time.



Default Transaction Thresholds can be viewed here:

The figure shows the 'Slow Transaction Thresholds' configuration page in the AppDynamics Pro interface. The left sidebar has a 'Navigate Here' callout pointing to the 'Slow Transaction Thresholds' link under 'Analyze'. The main panel shows 'User Transaction Thresholds' with sections for 'Slow Transaction Thresholds', 'Very Slow Transaction Thresholds', and 'Stall Threshold'. Red arrows point to the configuration fields for each threshold type. At the bottom, there are buttons for 'Apply to all Existing Business Transactions' and 'Save Default Slow Transactions Thresholds'.

If you want to change the thresholds for all or individual transactions see the transaction threshold policy configurations (see below). See [Thresholds](#).

The screenshot shows the 'Set Individual Business Transaction Thresholds' section for the '/httpTo3d' transaction. It includes three threshold types: Slow Transaction Threshold (set to 'Greater than 3 Standard Deviations for the last 2 hours'), Very Slow Transaction Threshold (set to 'Greater than 4 Standard Deviations for the last 2 hours'), and Stall Threshold (set to 'Stall occurs when a transaction takes more than 45 seconds'). A 'Save Slow Transactions Thresholds' button is at the bottom.

To troubleshoot slow transactions, see [Troubleshoot Slow Response Times for Java](#).

By Default, errors are determined when HTTP Error Codes are returned and by default AppDynamics instruments Java error and warning methods such as logger.warn and logger.error. AppDynamics captures the exception stack trace and automatically correlates it with the request. To learn how to change this, such as to reduce the number of errors reported by AppDynamics by default or to add redirect error pages, see [Configure Error Detection](#).

## Troubleshooting Tutorials for Java

### Tutorial for Java - Business Transaction Health Drilldown



Business Transaction Drilldown

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

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

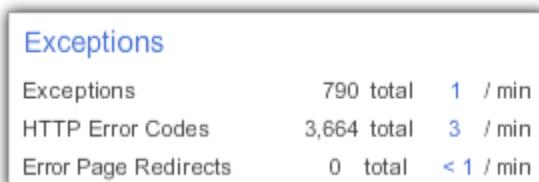
### Tutorial for Java - Exceptions

- [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](#)

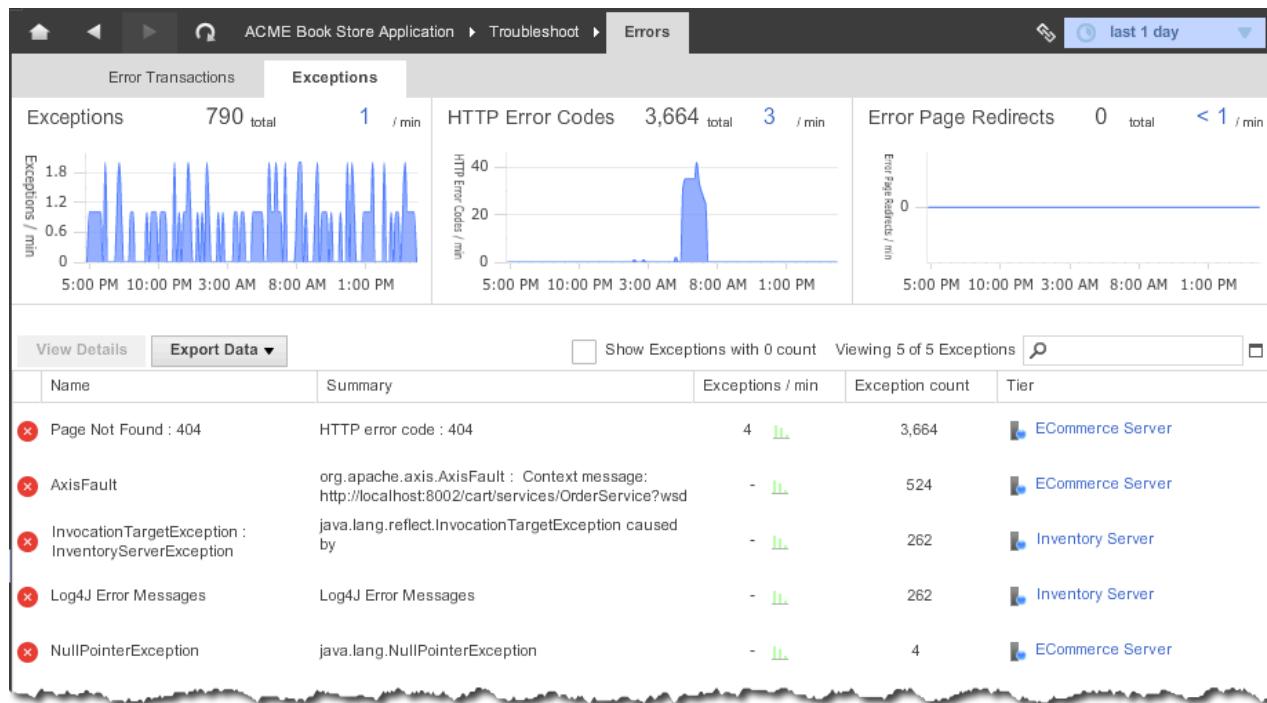
### The Exceptions

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.



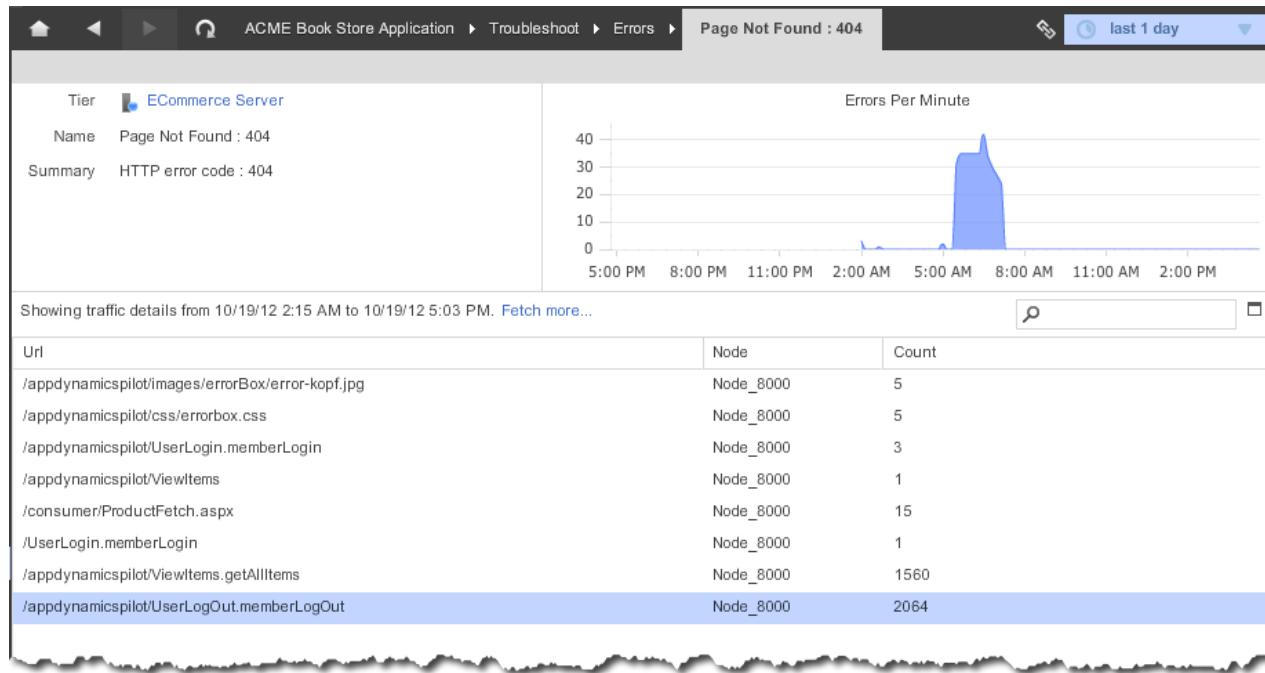
Click Exceptions to quickly see a list, ordered by frequency.



### 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.

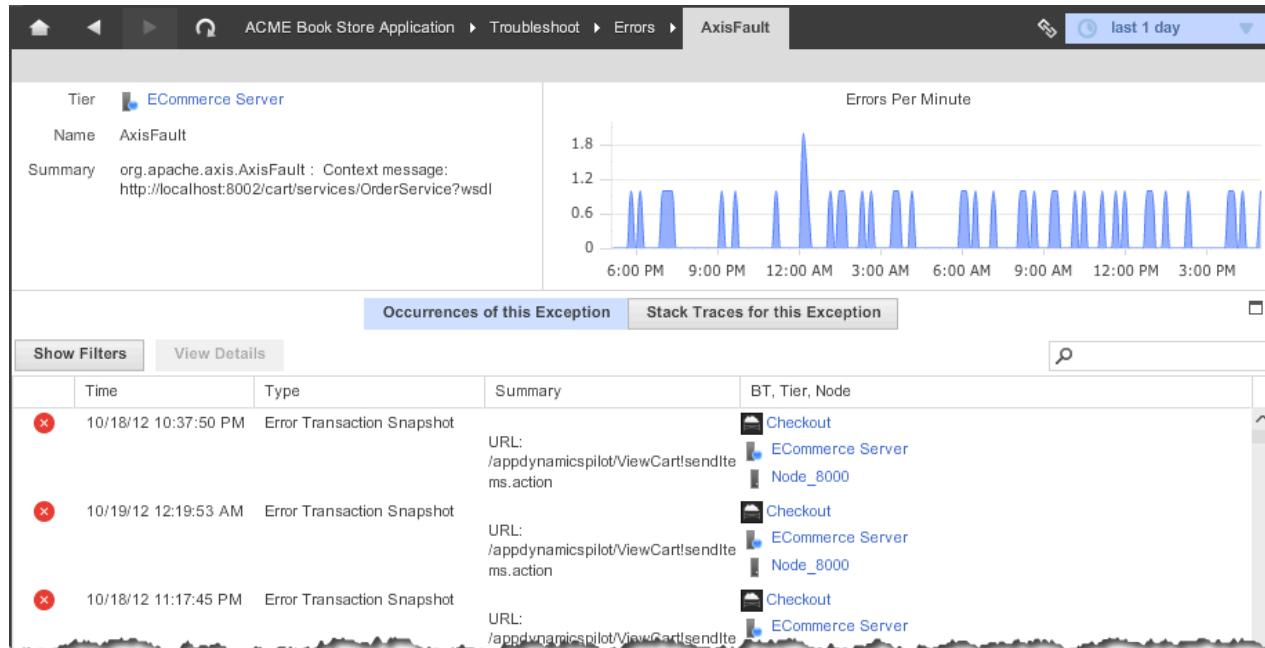
To find out more about the 404 error, click the row.



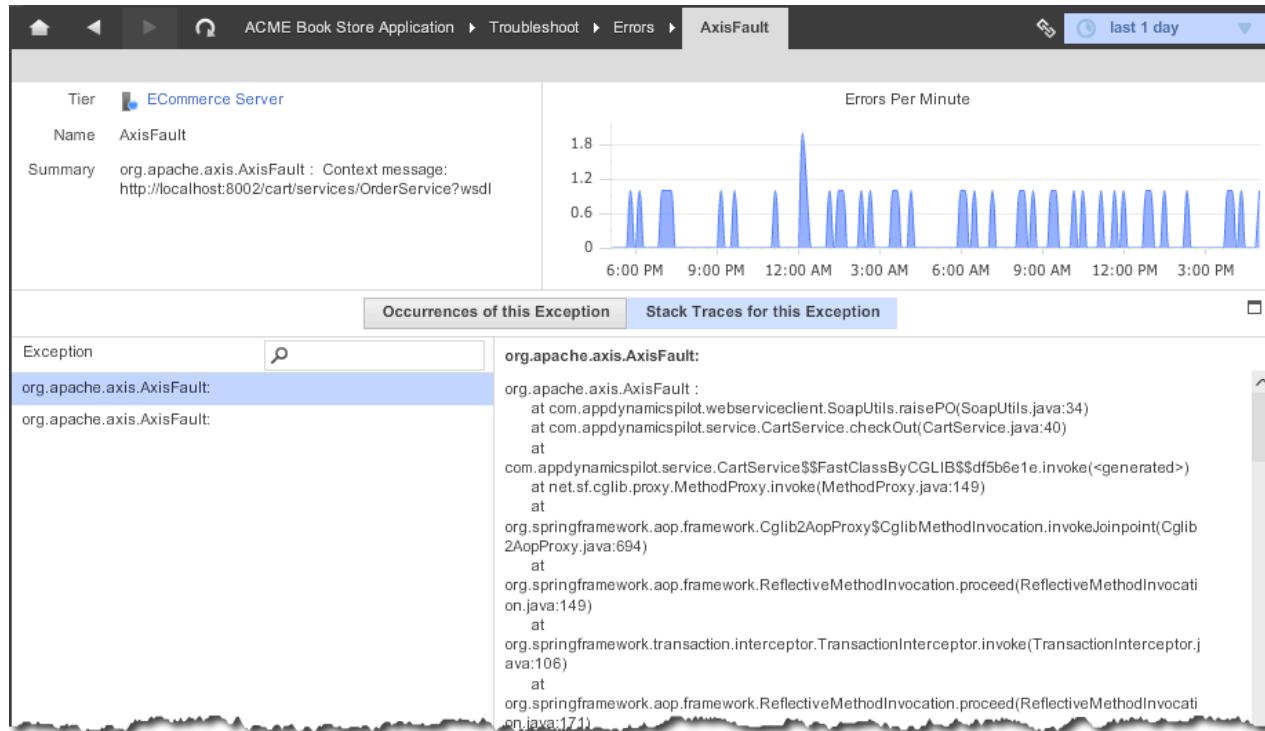
The list of URLs shows pages that have 404 errors. The memberLogOut and getAllItems URLs have the most 404 errors. You can provide this information to the web team to determine why those pages have so many 404 errors.

### 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.



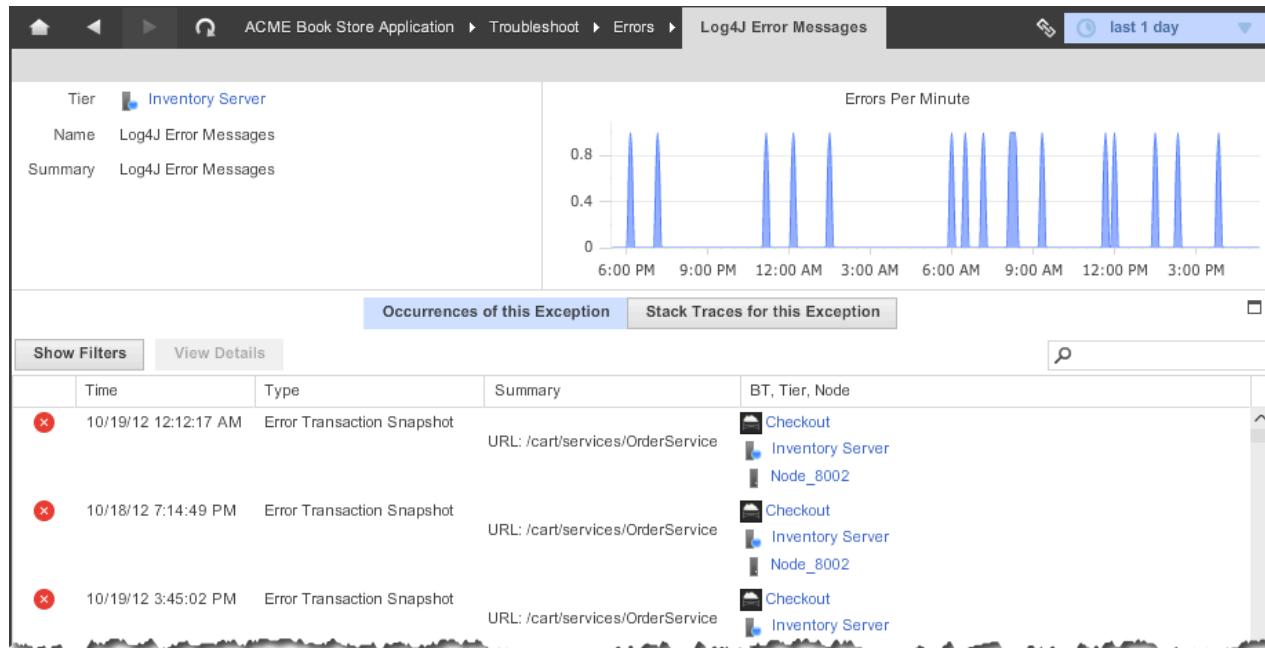
Click a row and then click the Stack Traces for This Exception tab to drill down further. Then click on one of the exceptions to see the stack trace.



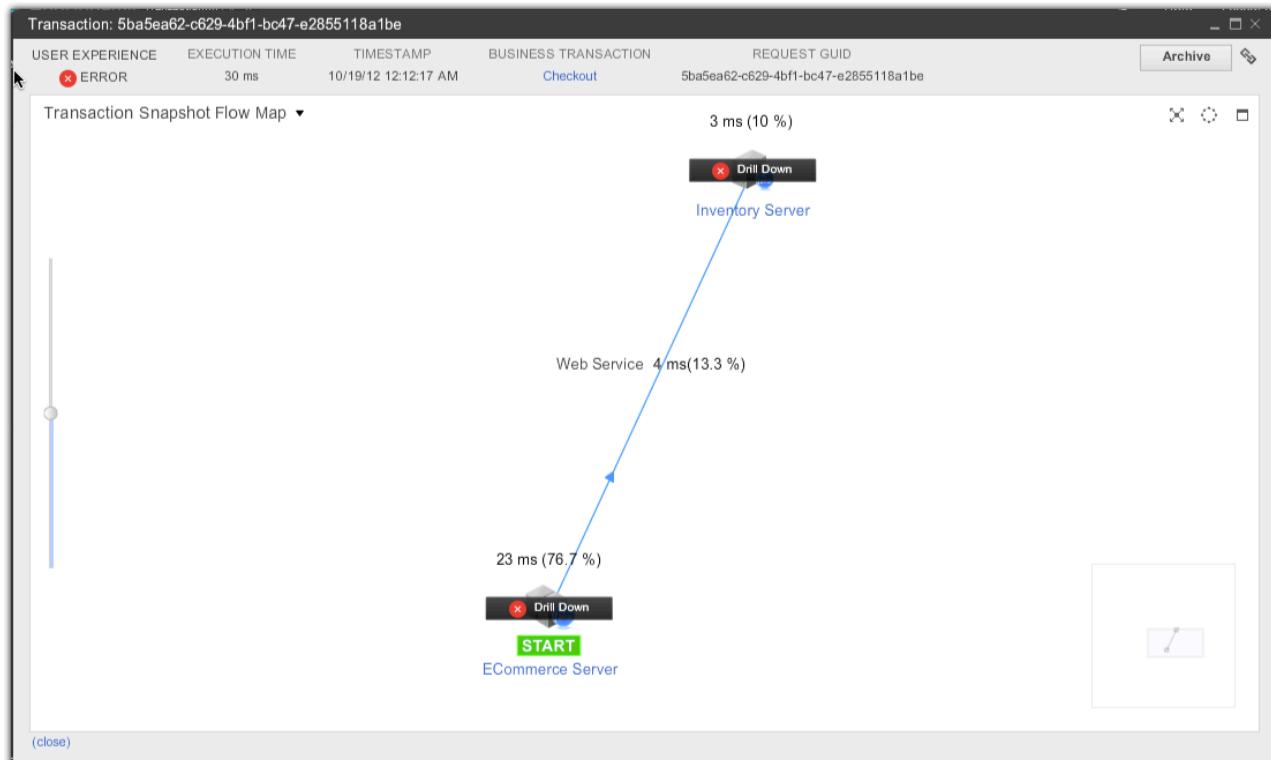
Share the stack trace with the development team to solve the problem.

### 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.

The screenshot shows the 'Call Drill Down' window for the same transaction. The left sidebar lists categories: SUMMARY, SQL CALLS, HTTP PARAMS, COOKIES, USER DATA, ERROR DETAILS, HARDWARE / MEM, NODE PROBLEMS, and ADDITIONAL DATA. The SUMMARY section displays the following details:

- 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)
- Error 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 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

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

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 AppDynamics Call Drill Down interface. The title bar reads "Call Drill Down. Exe Time: 3 ms Timestamp: 10/19/12 12:12:17 AM BT: Checkout GUID: 5ba5ea62-c629-4bf1-bc47-e2855118a1be". The left sidebar has a "SUMMARY" tab selected, showing metrics like User Experience (ERROR), Execution Time (3 ms), and CPU Time (0 ms). The main pane displays an error message: "[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 - com.appdynamics.inventory.OrderService : Error in creating order5". Below the error message, it shows the transaction path: "Inventory Server / Node\_8002 / Checkout". Other details include URL (/cart/services/OrderService), Session ID (not found), User Principal (No User Principal), Process ID (8153), Thread Name (http-8002-Processor18), and Thread ID (46). Transaction Thresholds show Slow and Very Slow thresholds. The Request GUID is 5ba5ea62-c629-4bf1-bc47-e2855118a1be. At the bottom left is an "Export to PDF" button, and at the bottom center is 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**.

**ACME Book Store Application**

**Business Transactions**

**Servers**

- App Servers
  - ECommerce Server
    - Node\_8000
    - Node\_8003
  - Inventory Server
  - Order Processing Server

Databases

Remote Services

Events

End User Experience

Troubleshoot

- Slow Response Times
- Errors
- Policy Violations

Analyze

Configure

- Instrumentation
- Policies
- Alerts
- Slow Transaction Thresholds

Instrumentation

Save Error Configuration

**Error Detection Using Logged Exceptions or Messages**

**Define where AppDynamics will look for log messages or exceptions to detect errors**

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

Configure AppDynamics to look for logged errors or method invocation. This can be used if you use a log4j or java.util.logging.

Name

Add Custom Logger Definition Edit

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

**Ignored Exceptions**  
Ignore these exceptions when detecting errors

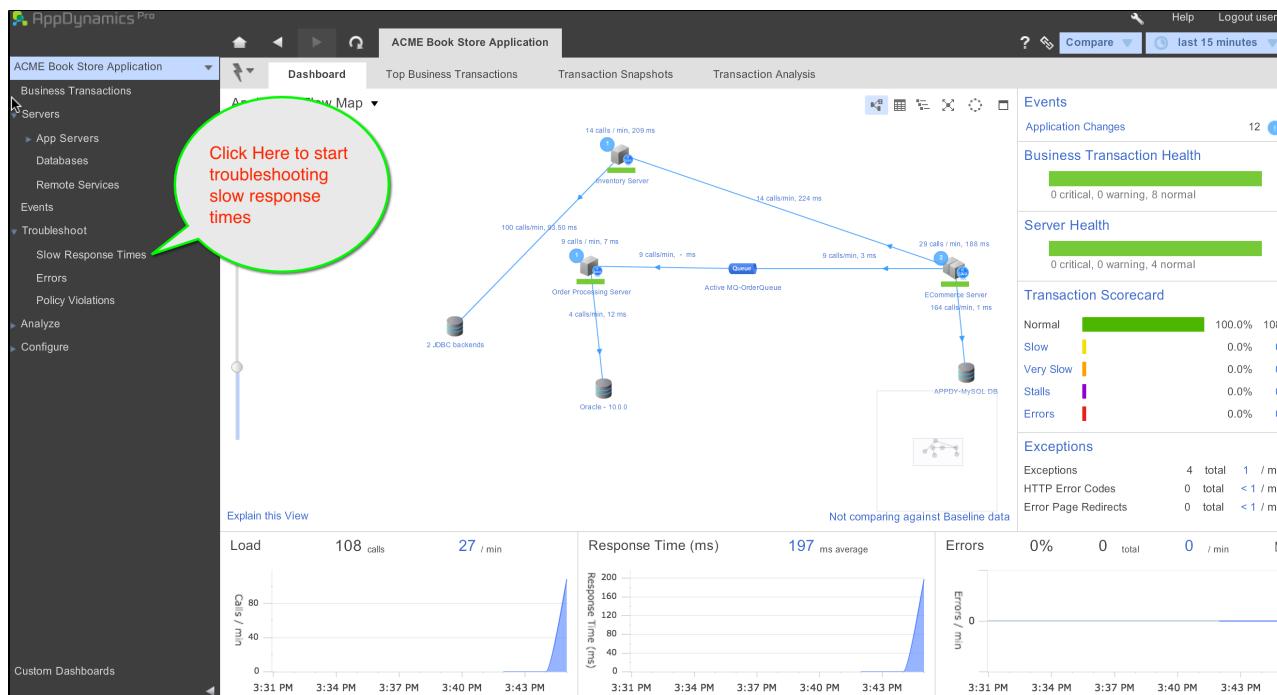
**Ignored Messages**  
Ignore these logged messages

## Learn More

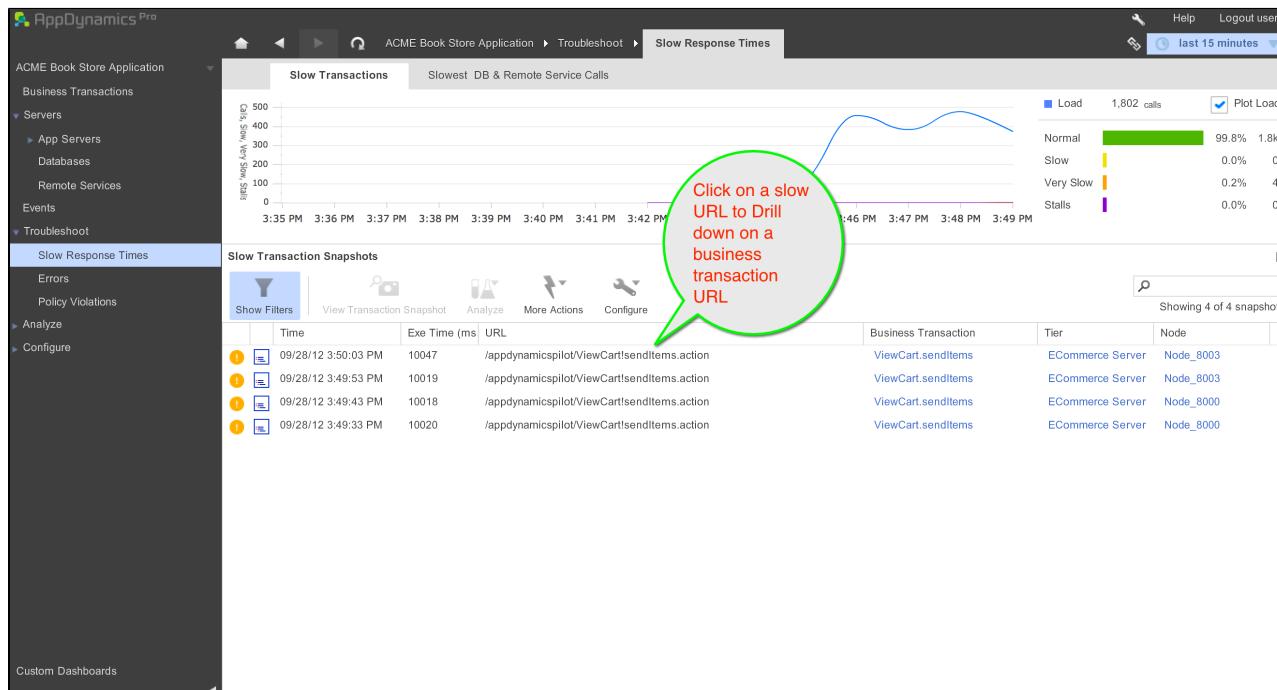
- Troubleshoot Errors
- Configure Error Detection

## Tutorial for Java - Slow Transactions

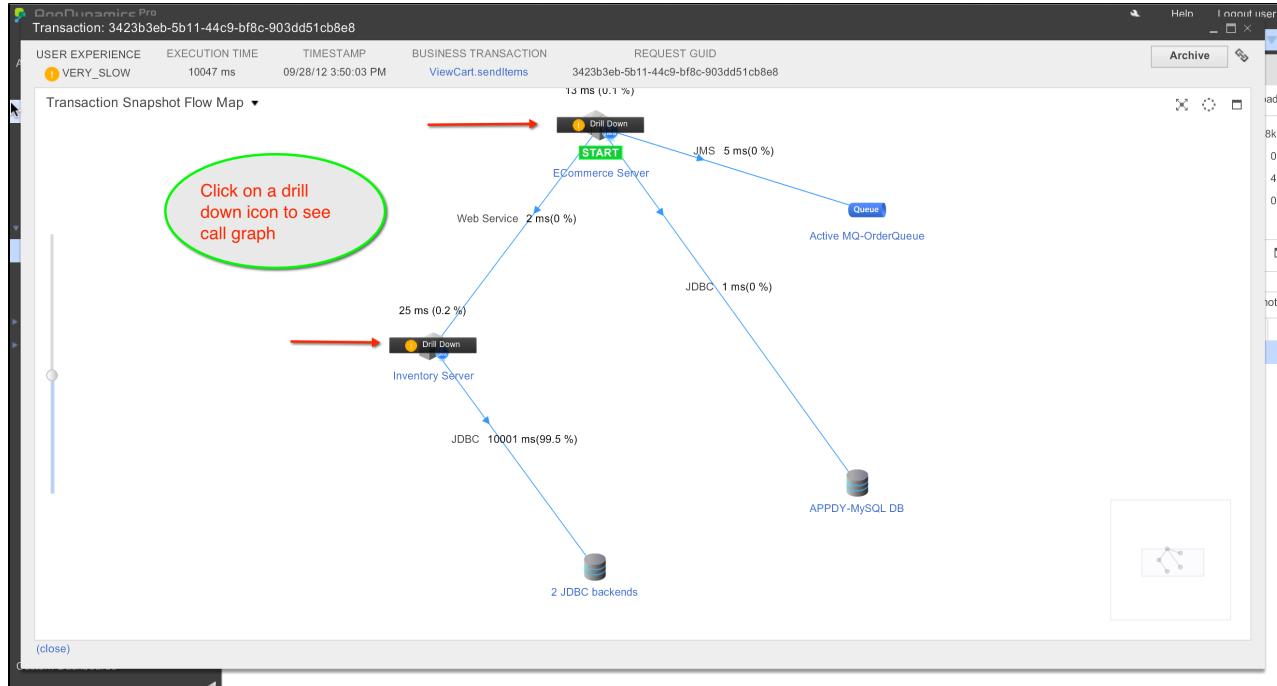
To begin troubleshooting, click **Troubleshoot -> Slow Response Times**:



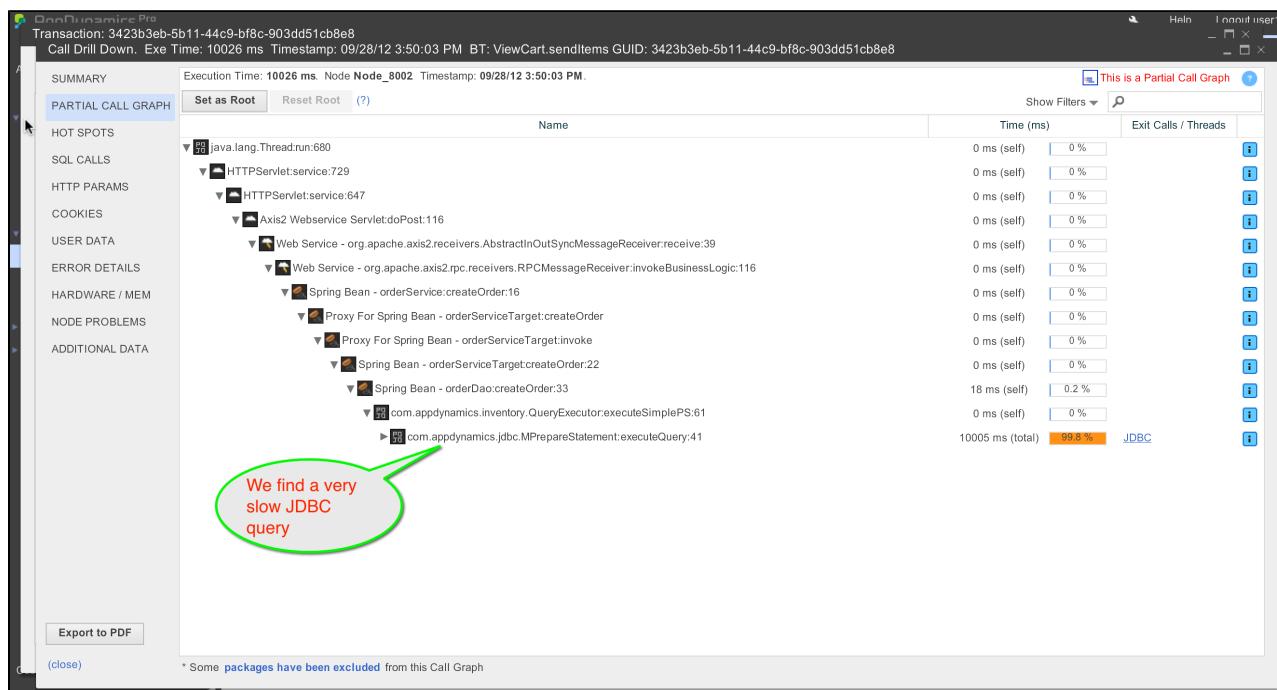
To troubleshoot slow business transaction URLs, select the Slow Transactions tab and use the Transaction Snapshots pane:



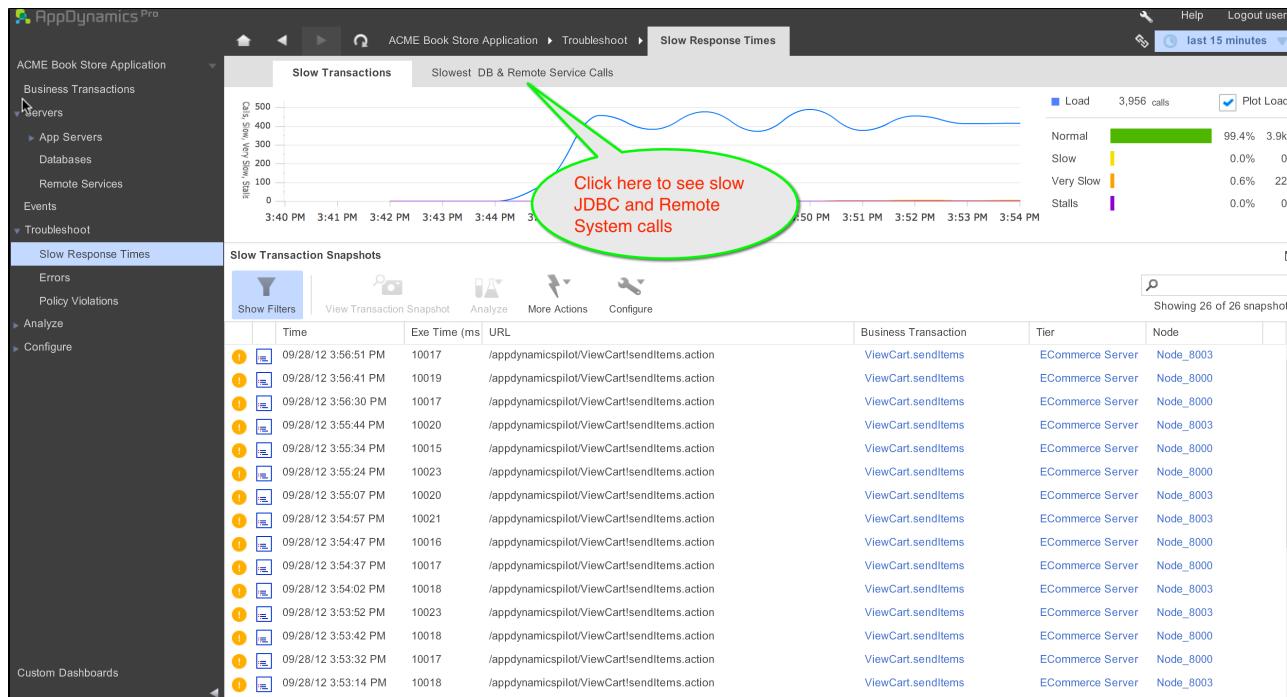
Once you select the URL you will see a visualization of the transaction. You can drill into a call graph by clicking the drill-down icon.



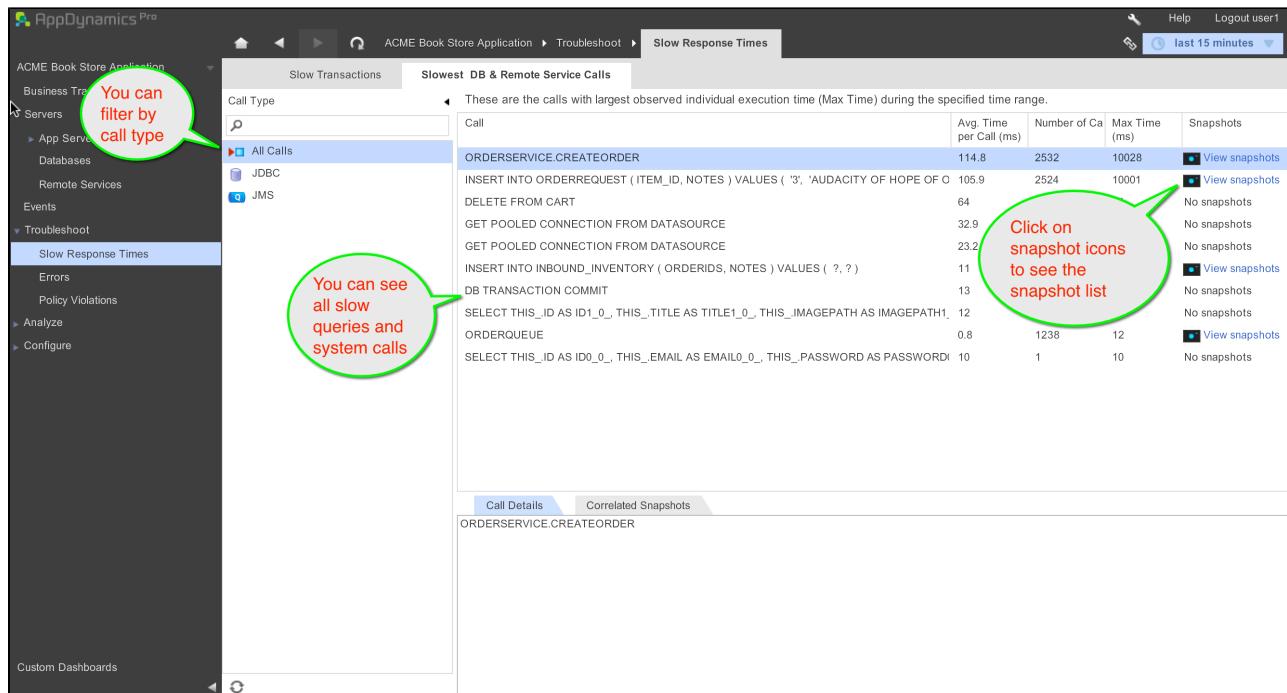
Once you are in the call graph you can look for methods that have a significant response time. For example, the executeQuery method is responsible for 99% of response time:



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)

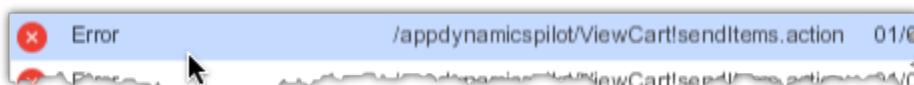
### How to Investigate

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

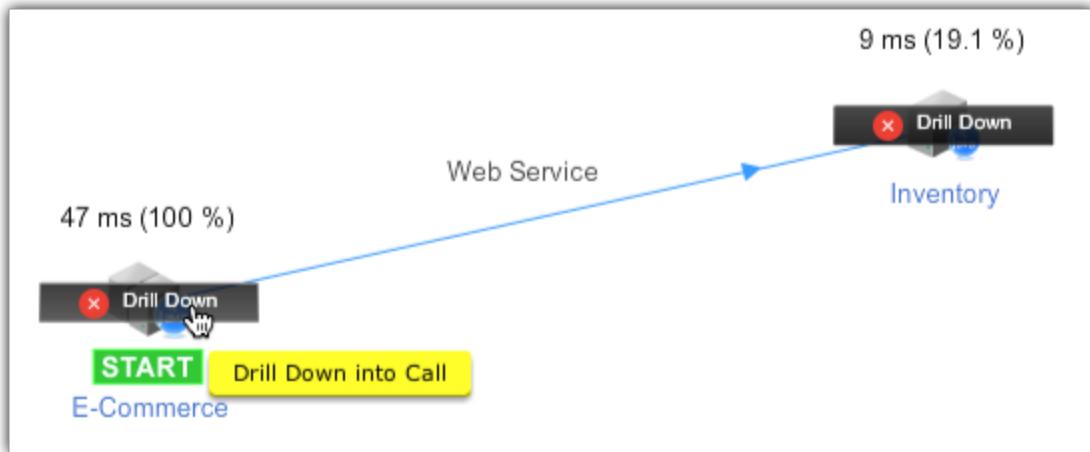
#### Investigating Errors

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**

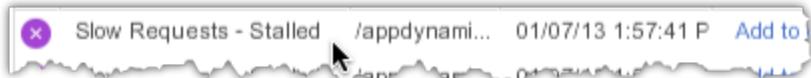
SUMMARY	User Experience
SQL CALLS	ERROR
HTTP PARAMS	Execution Time 47 ms
COOKIES	CPU Time 0 ms 0 %
USER DATA	Transaction Timestamp 01/07/13 1:58:32 PM (server) 01/07/13 1:58:32 PM (agent)
ERROR DETAILS	Summary [Error] - com.appdynamicspilot.webserviceclient.SoapUtils::There was an error invoke service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

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

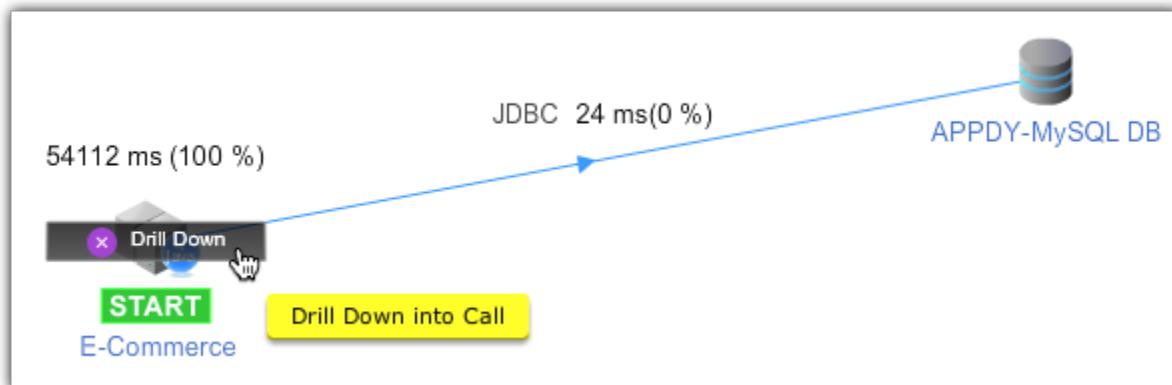
#### Investigating Stalled Business Transactions

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.

The Call Drill Down window is shown with the 'Summary' tab selected. Key details include:

- 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.appdynamicspilot.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(DefaultWorkflow
com.opensymphony.xwork2.interceptor.MethodFilterInterceptor.intercept(MethodFilterIntercept

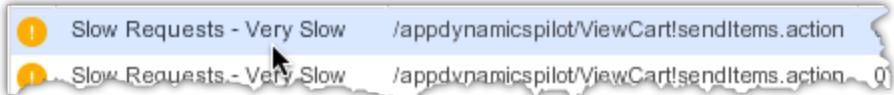
```

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

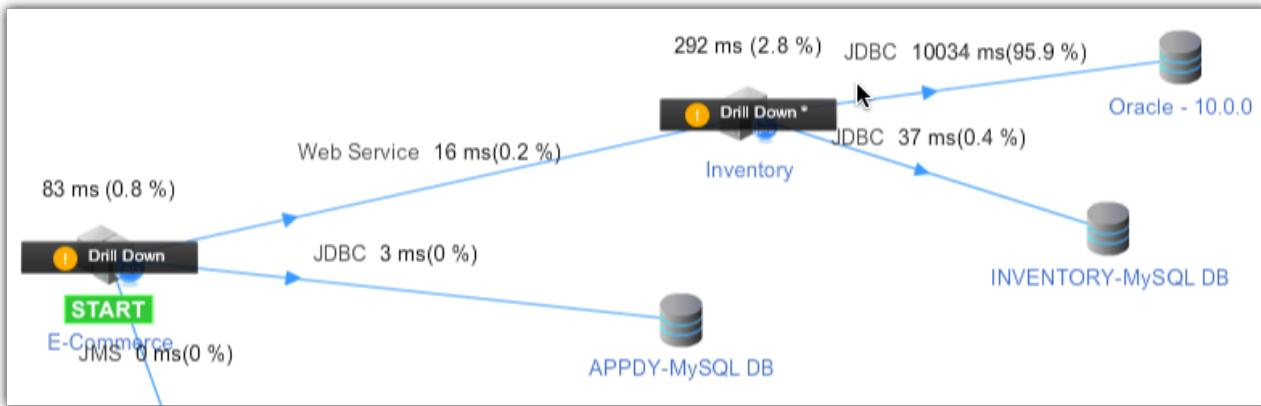
#### Investigating Slow Business Transactions

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.

	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 Ty	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**.



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.

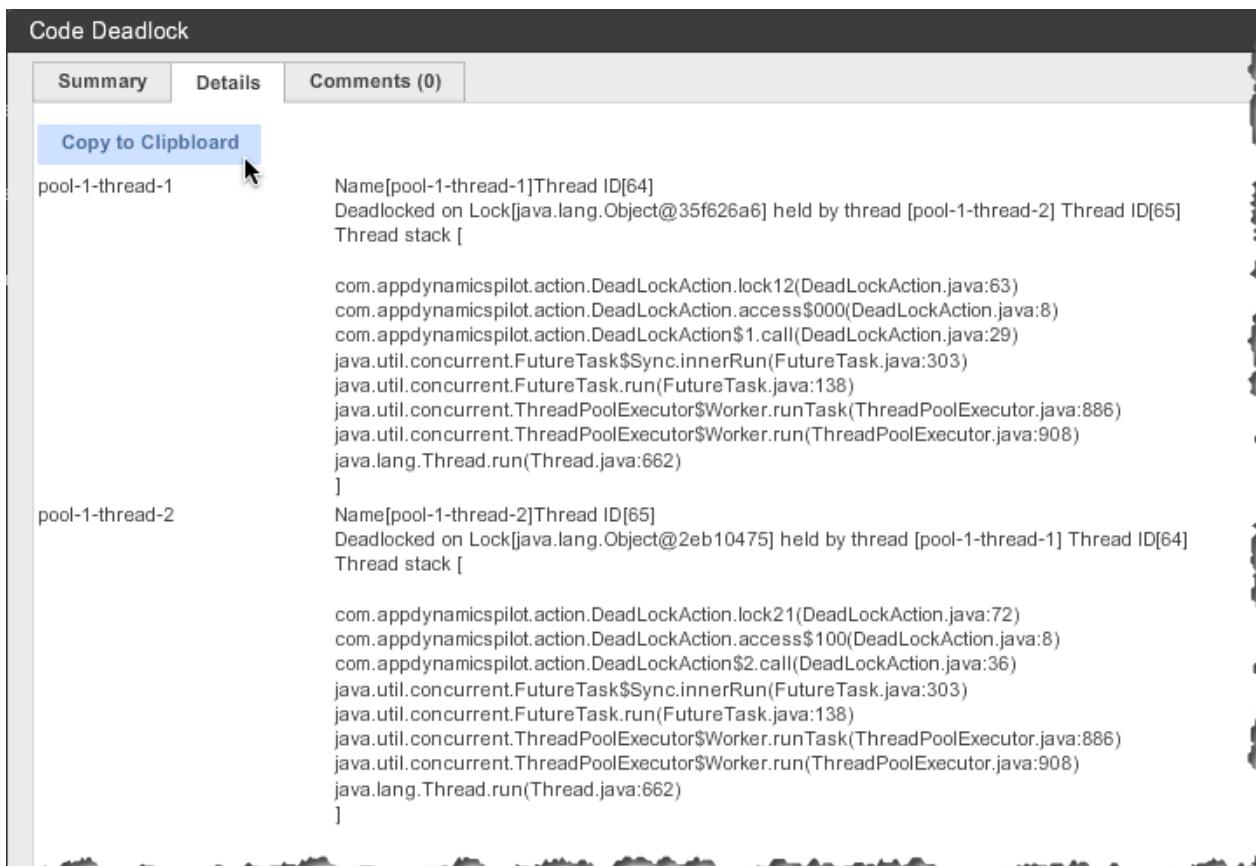
#### Investigating Code Deadlocks

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

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



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



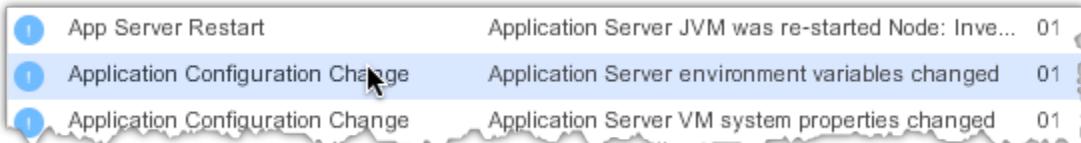
3. Use this information to troubleshoot code deadlock issues. Use the **Copy to Clipboard** button to save the deadlock details.

#### 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:



The screenshot shows a list of application change events. The first event is 'App Server Restart' with the description 'Application Server JVM was re-started Node: Inve... 01'. The second event is 'Application Configuration Change' with the description 'Application Server environment variables changed 01'. The third event is 'Application Configuration Change' with the description 'Application Server VM system properties changed 01'. The second event is highlighted with a blue background.

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



The screenshot shows the 'Application Configuration Change' details page. At the top, there are tabs for 'Summary', 'Details', and 'Comments (0)'. Below the tabs, there is a 'Copy to Clipboard' button with a cursor pointing at it. Underneath the button, it says 'Modified Variable 1: ACTION=start (changed to) ACTION=stop' and 'Modified Variable 2: \_EXECJAVA=start "Tomcat" "C:\Program Files\Java\jdk1.6.0\_33\bin\java" (changed to) \_EXECJAVA="C:\Prog...'.

## Tutorials for .NET

### Quick Tour of the User Interface



## Overview Tutorials for .NET

Quick Tour of the User Interface



Manual Installation and Configuration



## Tutorials for PHP

See also:

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

## First Time Using the App Agent for PHP

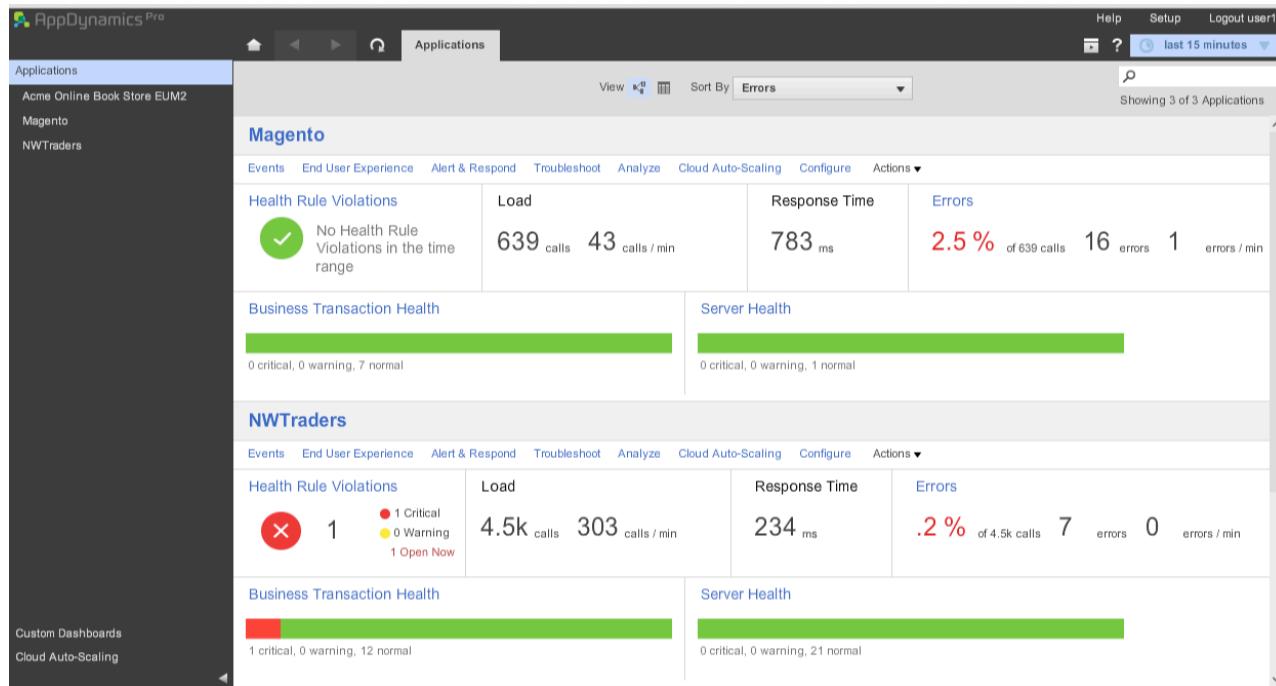
- [All Applications Dashboard](#)
- [Application Dashboard
  - Time Range
  - Flow Map and KPIs
  - Events
  - Transaction Scorecard
  - Exceptions and Errors](#)

This topic assumes that an application is already configured in AppDynamics and that you have already logged in to the AppDynamics Controller.

This topic is an overview of how AppDynamics detects actual and potential problems that users may experience while they are using your application - transactions that are slow, stalled or have errors. It helps you easily identify the root causes of these problems.

## All Applications Dashboard

When you log into the Controller you see the All Applications dashboard.



The All Applications dashboard shows high-level performance information about one or more business applications. Load, response time, and errors are standard metrics that AppDynamics calls "key performance indicators" or "KPIs". Other dashboard information includes:

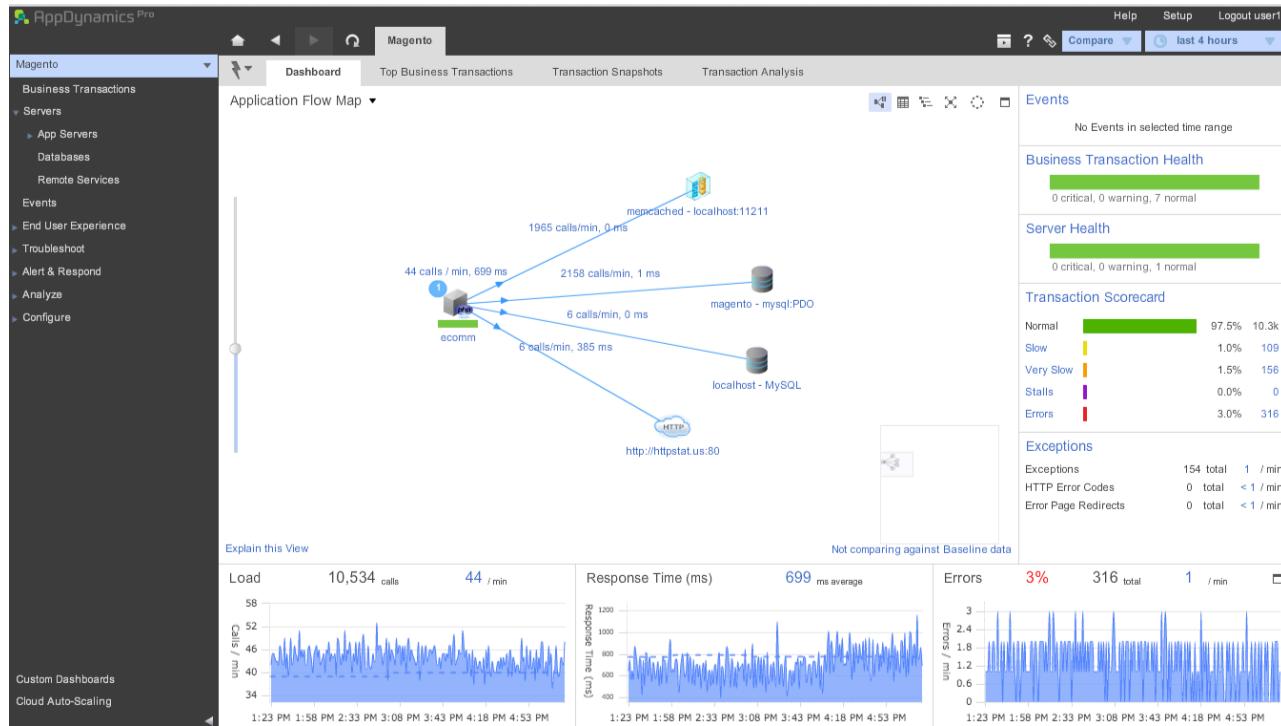
**Health Rule Violations:** AppDynamics lets you [define a health rule](#), which consists of a condition or a series of conditions based on metrics exceeding predefined thresholds or dynamic baselines. You can then use health rules in [policies](#) to automate optional remedial actions to take if the conditions trigger health rule violation events. AppDynamics also provides default health rules to help you get started.

**Business Transaction Health:** The health indicators are a visual summary of the extent to which a business transaction is experiencing critical and warning health rule violations. See [Troubleshoot Slow Response Times](#).

**Server Health:** Additional visual indicators track how well the server infrastructure is performing.

## Application Dashboard

Click an application to monitor, one that has some traffic running through it. The Application dashboard gives you a view of how well the application is performing.



This dashboard shows the performance of the Magento application for the last 4 hours. The flow map on the left displays your servers (application servers, databases, remote servers such as message queues, etc.) and metrics for the calls between them. Click, hold and move the icons around to rearrange the flow map. Use the scale slider and mini-map to change the view.

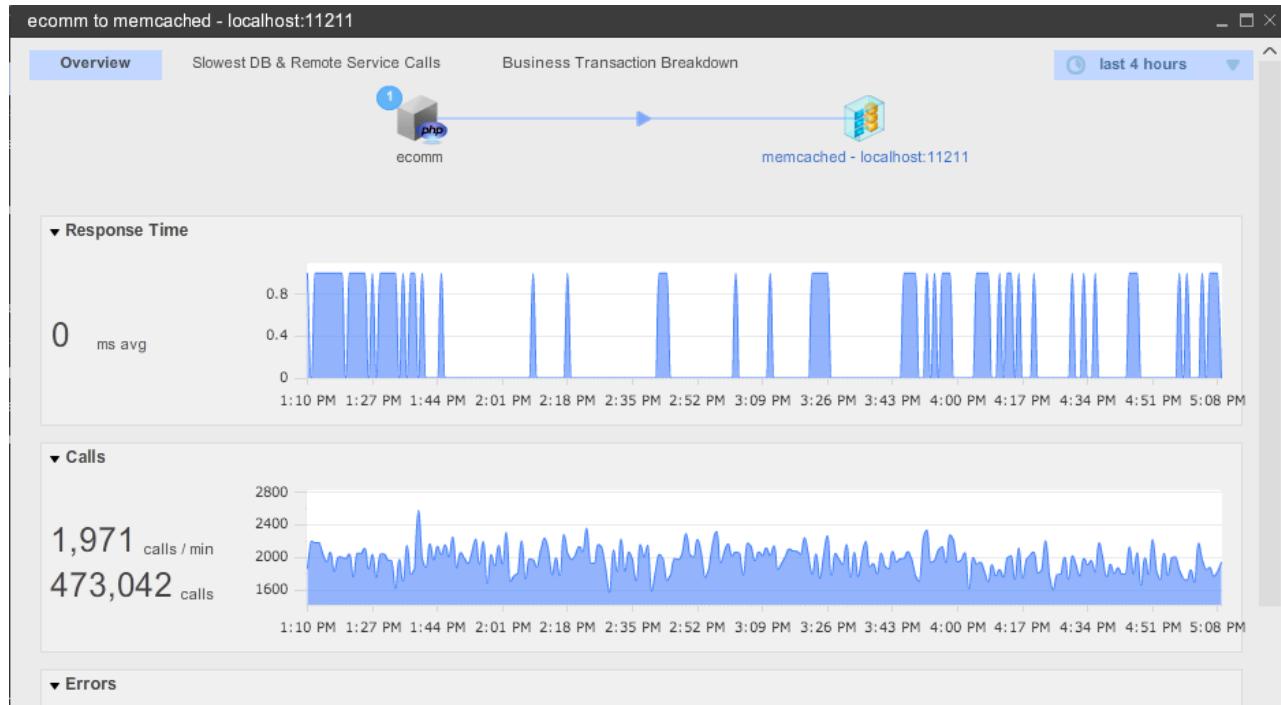
The trend graphs at the bottom of the dashboard show the key performance indicators over the selected time range for the entire application.

## Time Range

From the time range drop-down in the upper-right corner select the time range over which to monitor - the last 15 minutes, the last couple of hours, the last couple of days or weeks. Try a few different time ranges and see how the dashboard data changes.

## Flow Map and KPIs

In the flow map, click any of the blue lines to see more detail on the aggregated key performance metrics (load, average response time and errors) between two servers. For example, clicking the line from ecomm to memcached-localhost:11211 displays the following detail for that flow.



## Events

An event represents a change in application state. The Events pane lists the important events occurring in the application environment.

See [Monitor Events](#).

## Transaction Scorecard

The Transaction Scorecard panel shows metrics about business transactions within the specified time range, covering the percentage of instances that are normal, slow, very slow, stalled or have errors. Slow and very slow transactions have completed. Stalled transactions never completed or timed out. [Configurable thresholds](#) define the level of performance for the slow, very slow and stalled categories. See [Scorecards and Transaction Snapshots](#).

## Exceptions and Errors

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

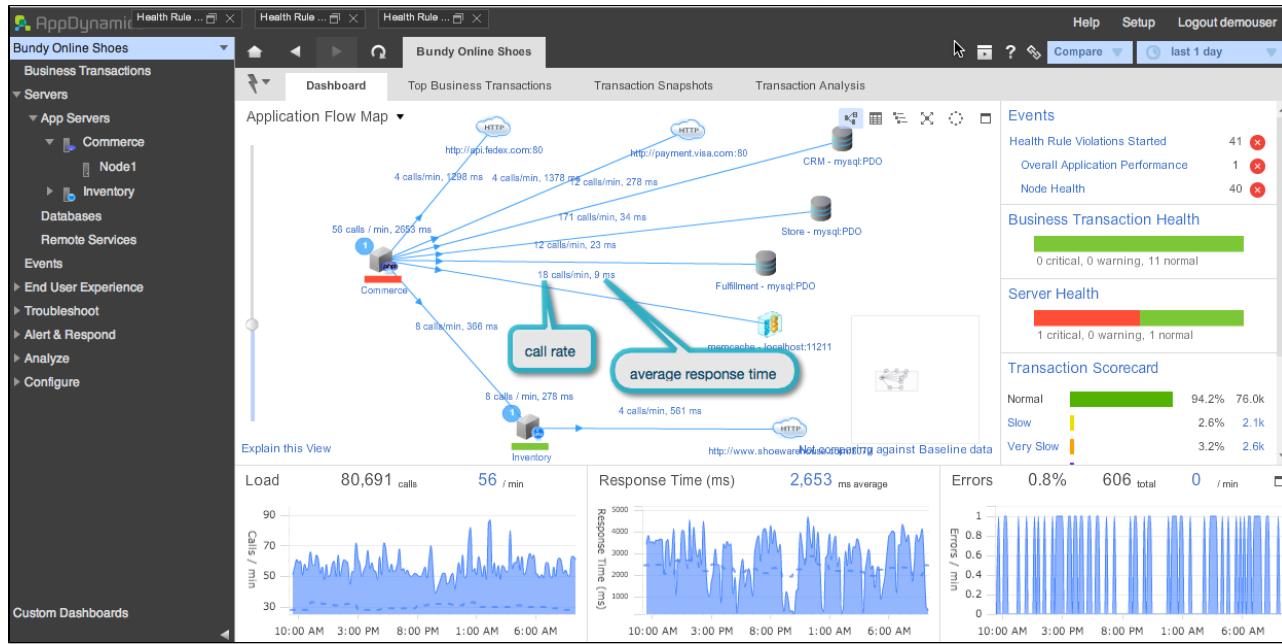
An error is a departure from the expected behavior of a business transaction, which prevents the transaction from working properly.

See [Troubleshoot Errors](#).

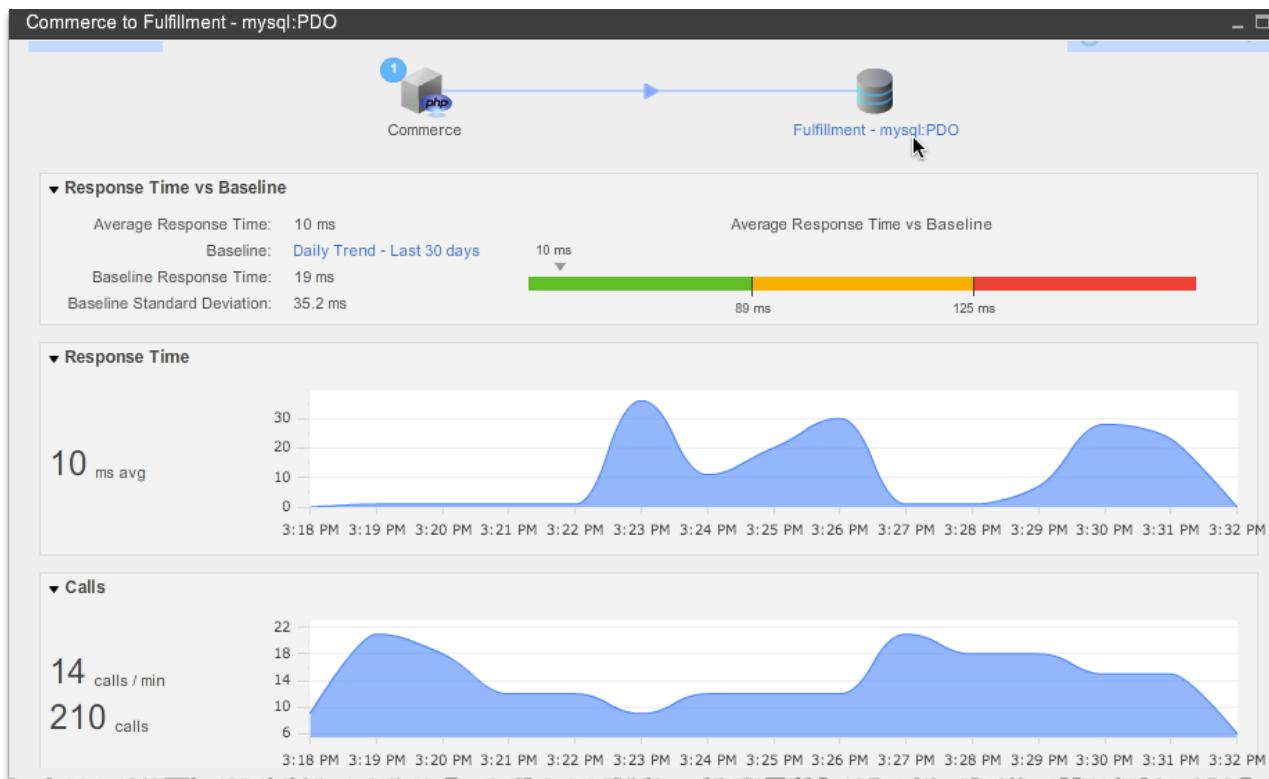
## Tutorial for PHP - Flow Maps

Flow maps graphically show the health of your application, all tiers, and the communication between the tiers. They include summary indicators such as call rates and the average response times the server takes to execute each flow.

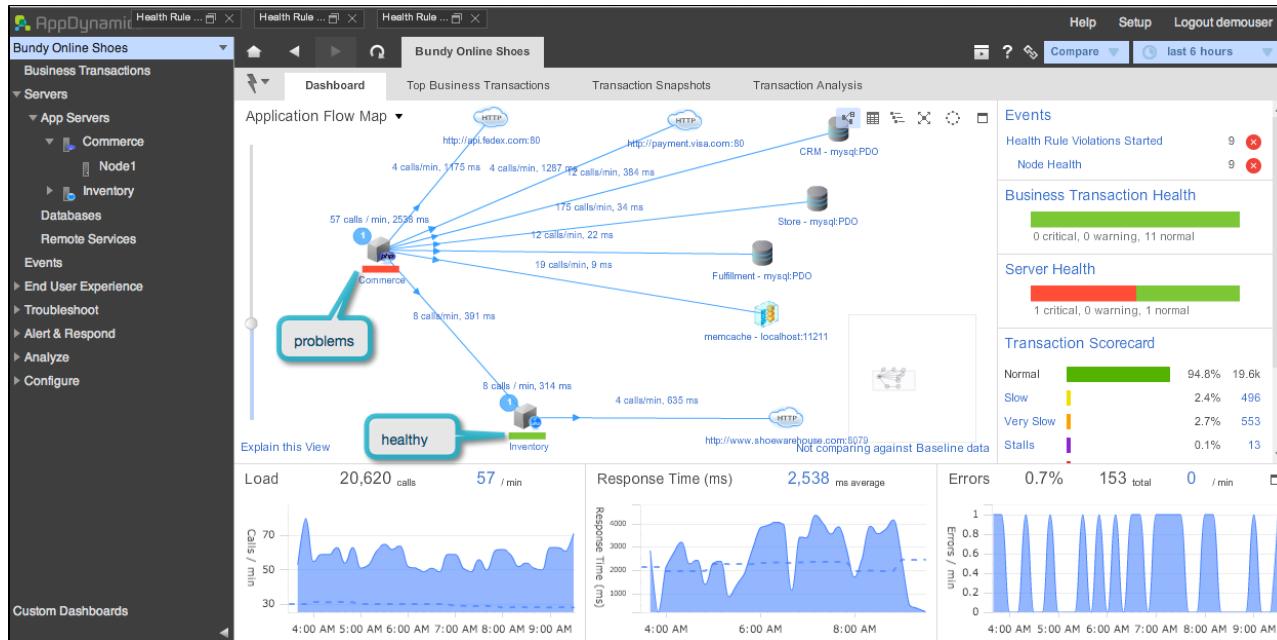
This is the application dashboard with its flow map for a PHP application.



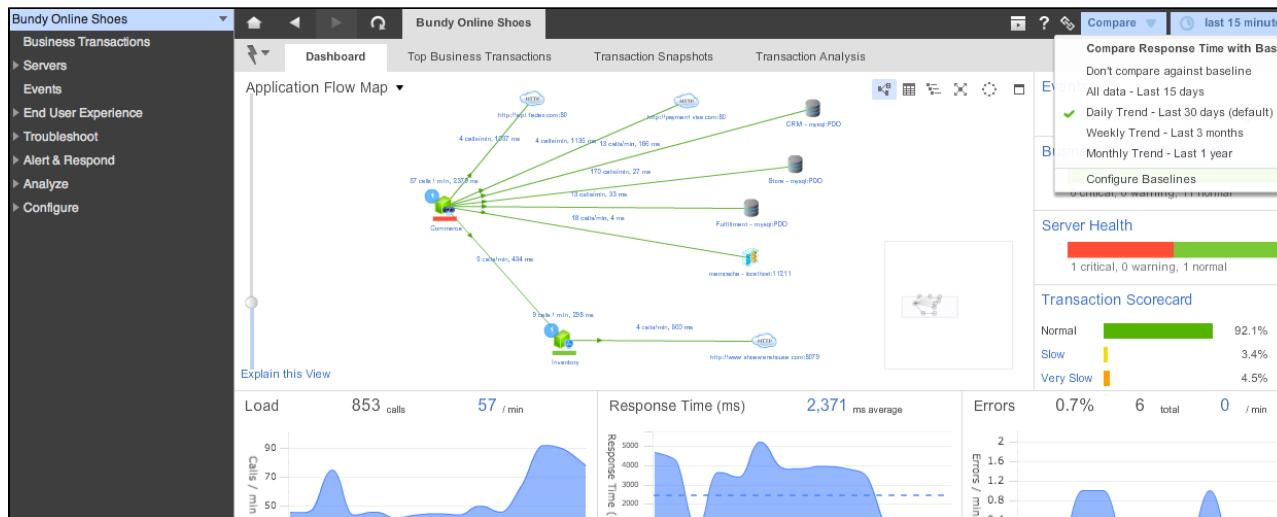
To see a summary of a single flow between two components, click the summary indicators along the line connecting the two components in the flow map. For example, the summary below displays the flow between the Commerce server and the Fulfillment-msq:PDO database.



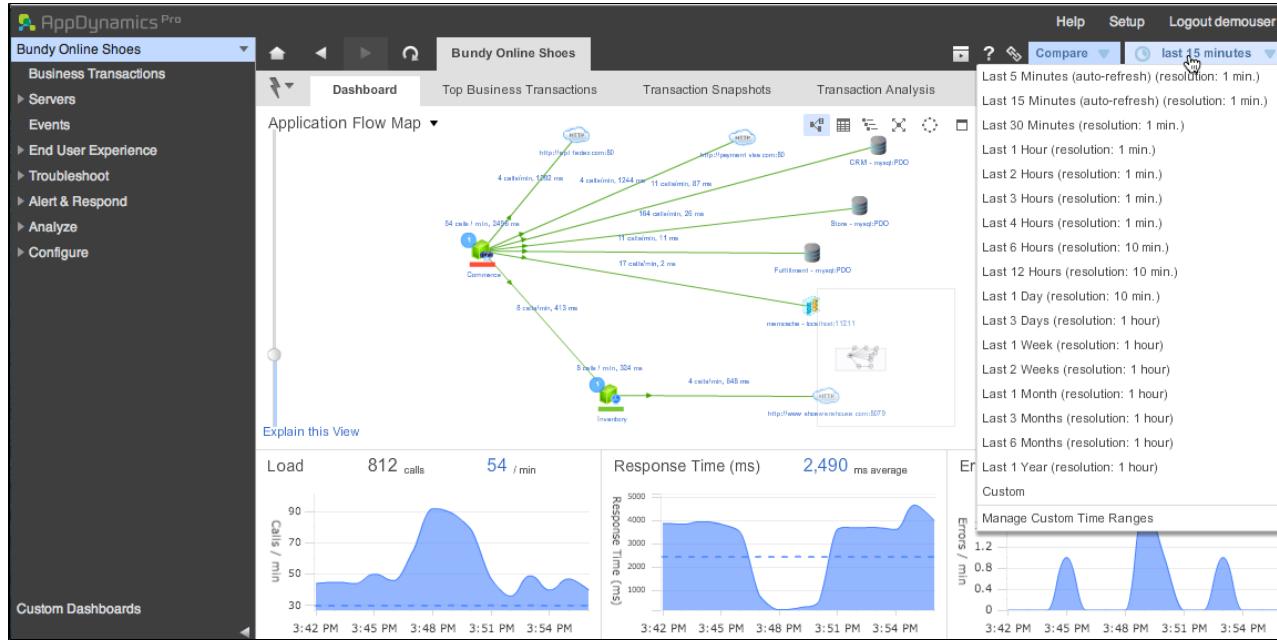
In the flow map, visual indicators alert you to tiers experiencing problems. The red bar under the Commerce tier indicates problems while the green bar under the Inventory tier indicates that this tier is healthy:



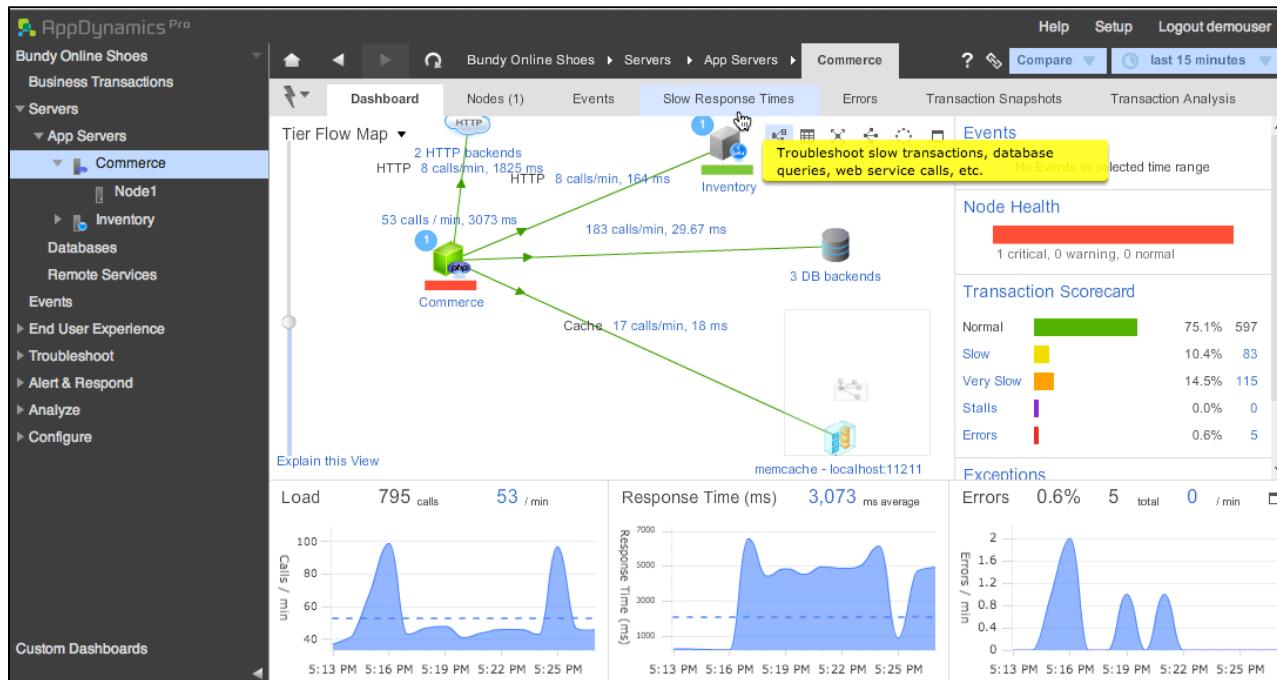
By default, the flow map computes tier health by comparing the state of the tier averaged over the last 15 minutes against the daily trend (the 30 day rolling average). You can change the time window for baseline comparison using the time window pull-down menu. You can also disable baseline comparisons. For an in-depth discussion of AppDynamics baselines see [Behavior Learning and Anomaly Detection](#).



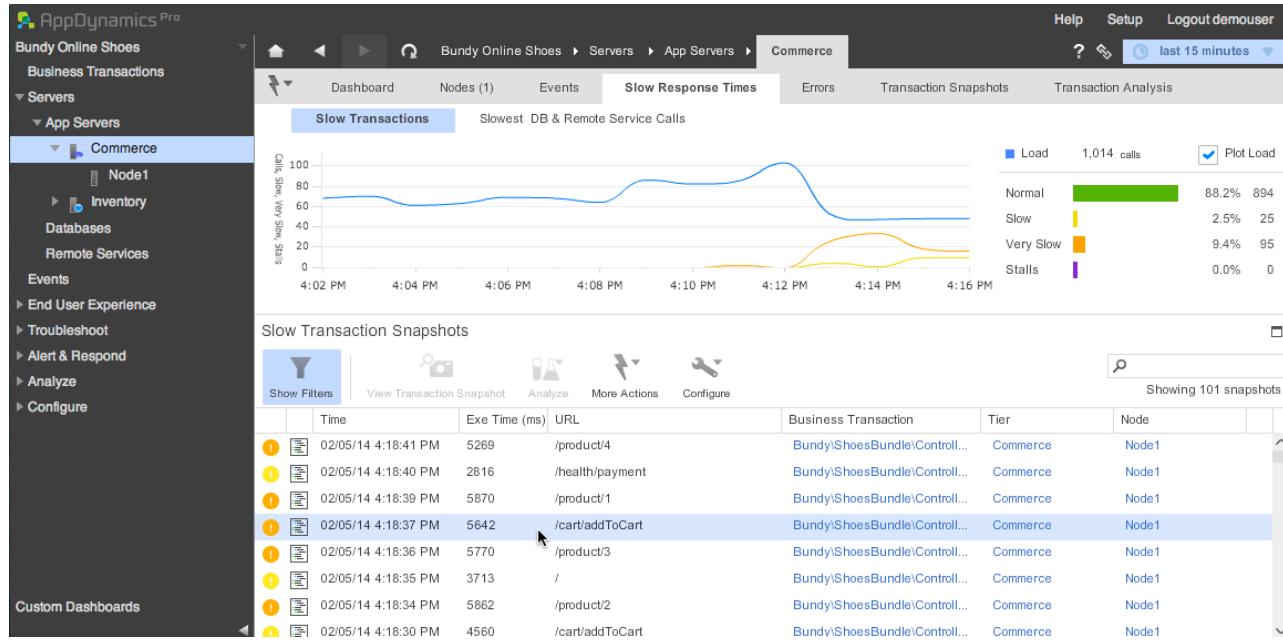
You can change the time range displayed in the flow map using the time window pull down menu. Changing the time range affects all the information in the dashboard and in all other dashboards for the application.



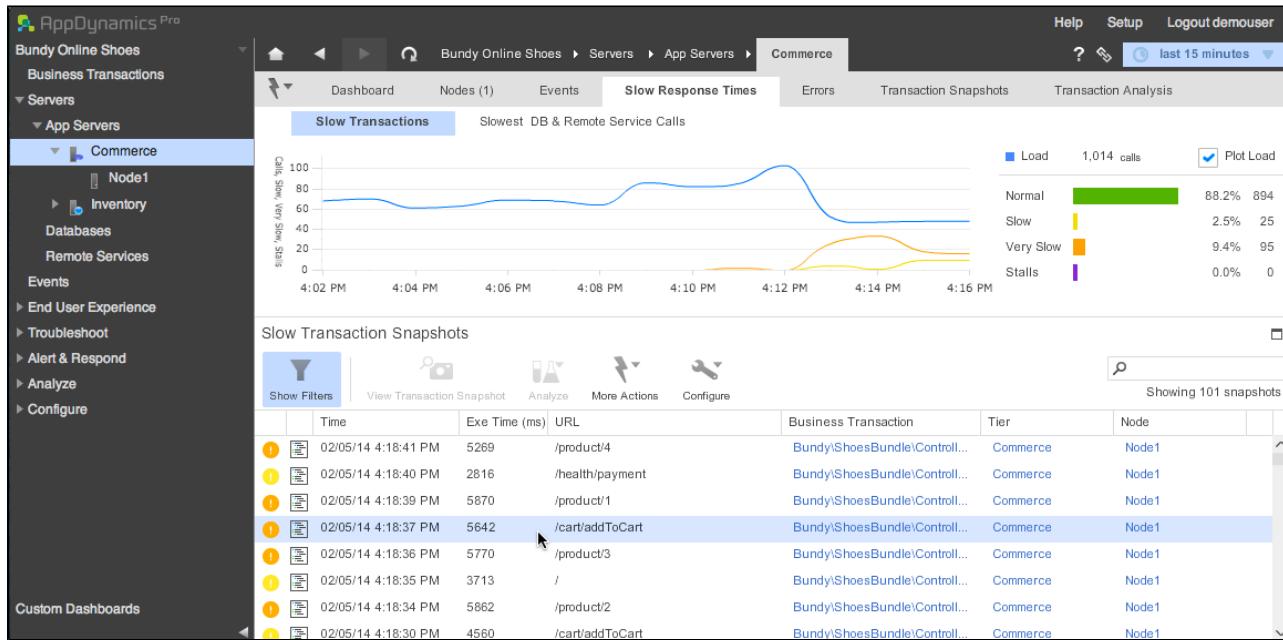
If a tier is experiencing problems, such as slow response times, click the tier name to see a flow map just of that tier:



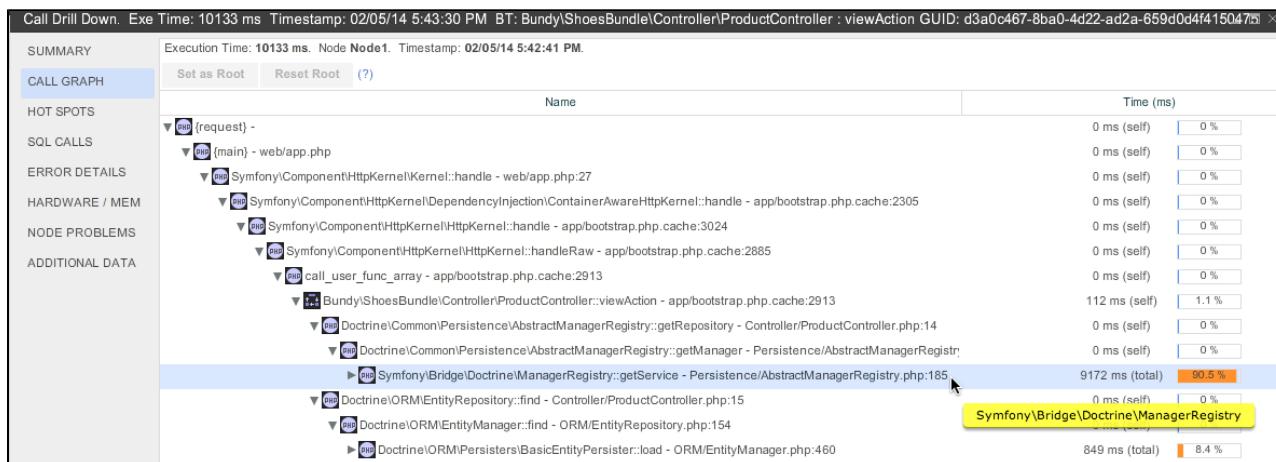
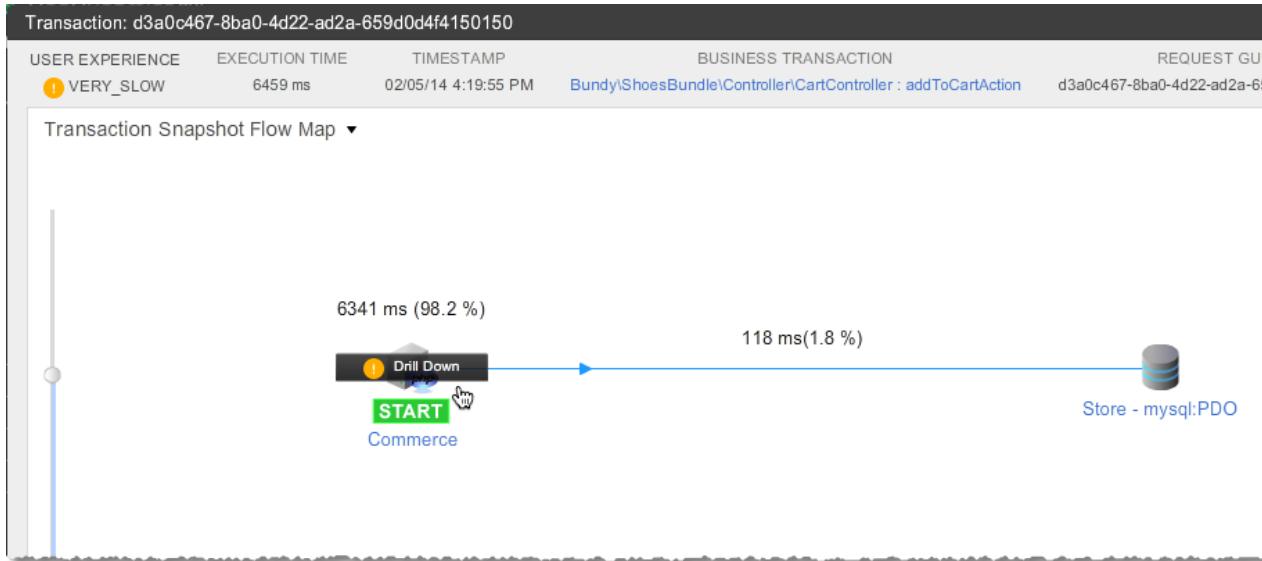
Click the Slow Response Times tab in the tier dashboard to view a graph of the slow response times on that server, optionally compared with the load (blue line), as well as a list snapshots of individual transactions that were slow.



From slow transaction snapshot list, double-click a slow transaction to drill down into the details of the causes of the slowdown:



From the transaction snapshot flowmap, click **Drill Down** to troubleshoot the slow transaction:



For more information see [Troubleshoot Slow Response Times for PHP and Transaction Snapshots](#).

## Tutorial for PHP - Server Health

- About PHP Server Health
- Viewing Health Rule Violations
- Modifying an Existing Health Rule
  - To access the node health rule configuration window
  - To modify the conditions that trigger an existing health rule violation
- Learn More

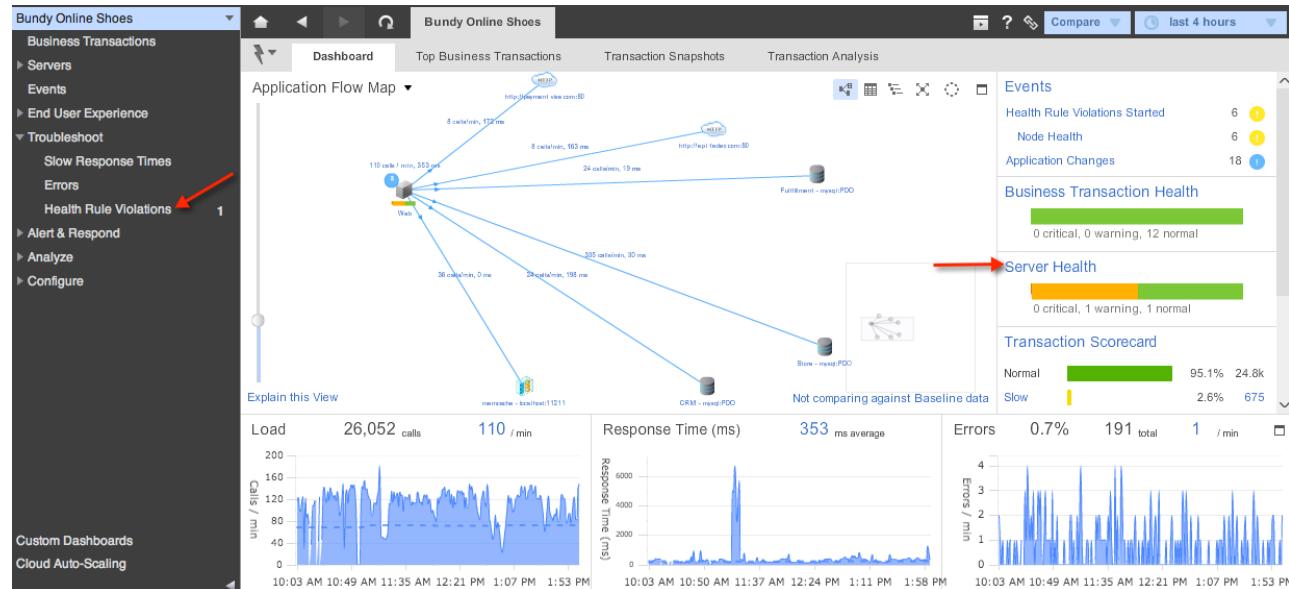
## About PHP Server Health

Health is an indicator of how actual performance compares to acceptable performance. Acceptable performance is defined by health rules which generate warning events for performance that may be and critical events for performance that definitely is of concern and requires investigation. A color other than green in the server health bar or in the icons in the Health tab in the app servers list indicates that health rule violation events exist.

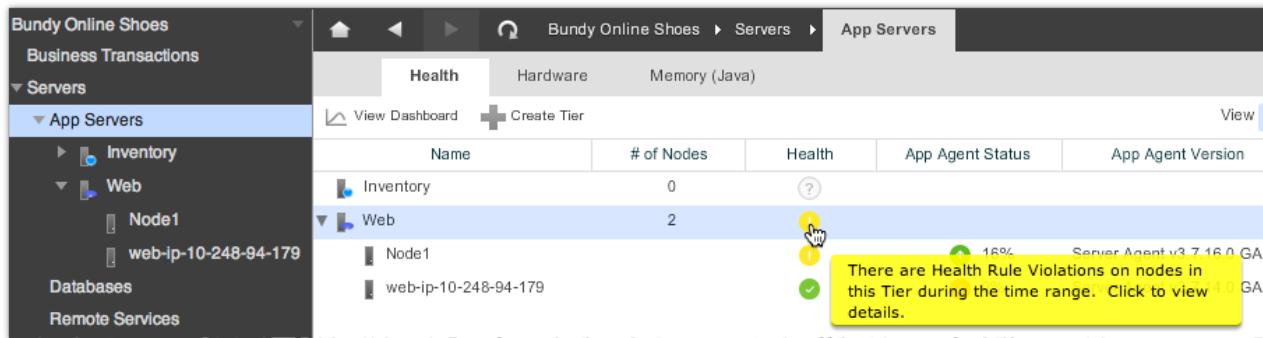
For server health, AppDynamics provides predefined default health rules for CPU utilization and physical memory utilization.

## Viewing Health Rule Violations

You can access information about health rule violations by clicking **Troubleshoot-> Health Rule Violations** in the left navigation pane or **Server Health** in the Server Health panel.



You can view the health rule violation details and the status of the violation:



Double-click on a health rule violation in the list to see details about the violation. From there you can view:

- the configuration for the health rule that was violated
- the node dashboard at the time of the health rule violation
- actions that were automatically executed (if any) in response to the health rule violation

**Health Rule Violation Started - Warning**

**Summary** Actions Executed (0)

Event Type: ! Health Rule Violation Started - Warning

Health Rule: ● Memory utilization is too high

Health Rule Type: Node Health - Hardware, JVM, CLR (cpu, heap, disk I/O, etc)

Affects: ■ Node1

Violation State: Open

Duration: Ongoing (3 hours, 35 minutes)  
From: 01/31/14 11:28:55 AM To: -

No Actions Executed [What does this mean?](#)

**Description**

Appdynamics has detected a problem with Node **Node1**.  
**Memory utilization is too high** started violating and is now **warning**.  
All of the following conditions were found to be violating  
For Node **Node1**:  
1) Hardware Resources|Memory|Used % Condition  
Used %'s value **76.0** was **greater than** the threshold **75** for the last **30** minutes

## Modifying an Existing Health Rule

Perhaps a predefined health rule is too restrictive, or not restrictive enough, for your application environment. For example the default health rule for CPU utilization triggers a warning event when a node exceeds 75% CPU utilization and a critical event when a node exceeds 90% utilization. You can modify these percentages as well as create your own health rules. Maybe the default setting is generating too many health rule violations and you want to change the configuration so that critical events are triggered when CPU utilization exceeds 95%.

### To access the node health rule configuration window

Do one of the following:

- If you currently viewing a violation in the Health Rule Violation window, click the link to the health rule.



- In the left navigation pane:
  1. Click **Alert & Respond->Health Rules**.
  - 2 Double-click the the health rule in the health rule list.

Name	Type	Enabled
Business Transaction response time is much higher than normal	Business Transaction Performance	✓
Business Transaction error rate is much higher than normal	Business Transaction Performance	✓
CPU utilization is too high	Node Health - Hardware, JVM, CLR	✓
Memory utilization is too high	Node Health - Hardware, JVM, CLR	✓

### To modify the conditions that trigger an existing health rule violation

1. In the Edit Health Rule window click the Critical Condition tab on the left.
2. Change the value in the specific value text field to another value.
3. Click **Save**.
4. In the Edit Health Rule window click the Warning Condition tab on the left.
5. Change the value in the specific value text field to another value.
6. Click **Save**.

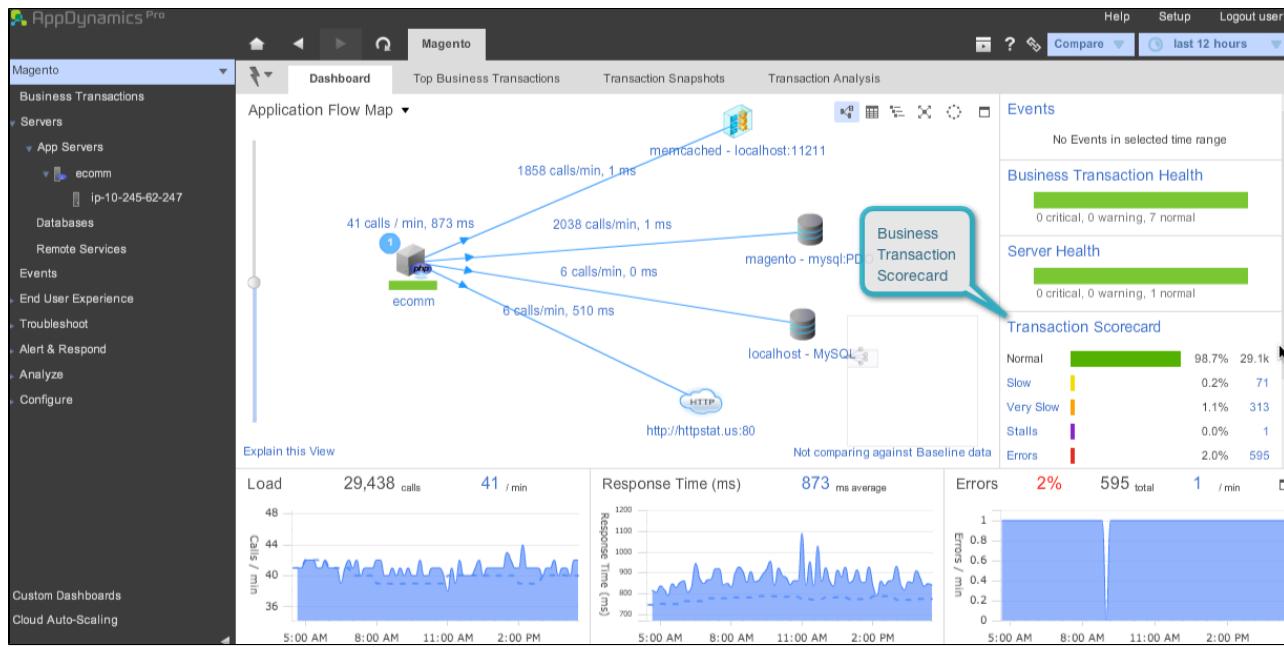
You can also add more conditions required to trigger the health rule violation by clicking **Add Condition**. Or change the times at which the health rule is in effect, in the Overview tab. Or restrict the nodes and tiers affected by the health rule in the Affects tab. Or create entirely new health rules from scratch. See [Health Rules](#) and [Configure Health Rules](#)..

## Learn More

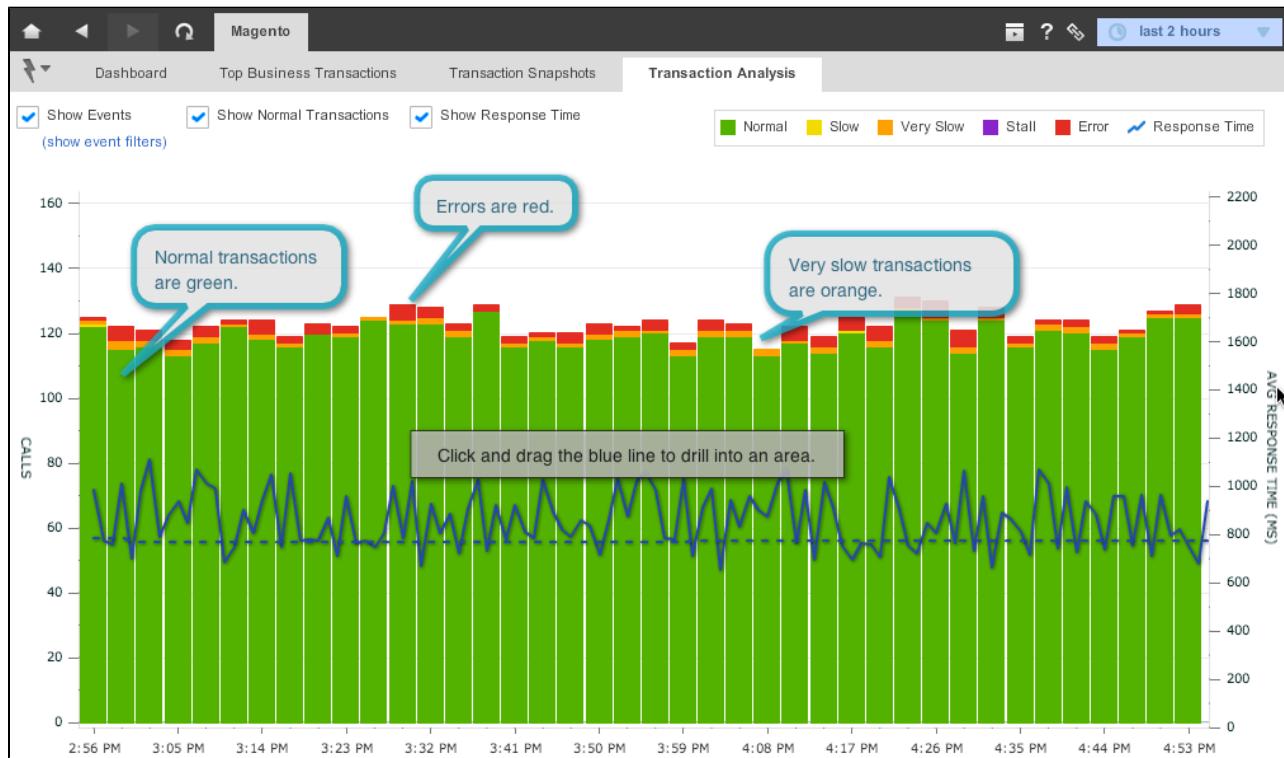
- [Health Rules](#)
- [Configure Health Rules](#)
- [Policies](#)
- [Actions](#)
- [Troubleshooting Server Health, AppDynamics in Action video](#)

## Tutorial for PHP - Transaction Scorecards

[Business transactions](#) are categorized as Normal, Slow, Very Slow, Stalls, or Errors in the transaction scorecard that appears in several dashboards. The categories are determined by configurable thresholds and by the AppDynamics error detection configuration.



To see the response time distribution over time, click the Transaction Scorecard link or the Transaction Analysis tab in a dashboard. For more information about this histogram see [Transaction Analysis Tab](#).



To view or modify the default thresholds, click **Configure->Slow Transaction Thresholds** in the left navigation pane.

User Transaction Thresholds

Background Tasks Thresholds

**Slow Transactions Thresholds**

**Slow Transaction Threshold**

- More than  % slower than the average of the last  hours
- Greater than  3 Standard Deviations for the last  hours

**Very Slow Transaction Threshold**

- More than  % slower than the average of the last  hours
- Greater than  4 Standard Deviations for the last  hours

**Stall Threshold**

- Disable Stall detection
- Stall occurs when a transaction takes more than  45 seconds.
- Stall occurs when a transaction's response time is  deviations above the average for the last 2 hours.

Thresholds can be static or dynamic; dynamic thresholds are based on historical data. You can configure what Slow, Very Slow, and Stall means for each business transaction in your application. For more information see [Thresholds](#).

User Transaction Thresholds

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

**Slow Transaction Thresholds**

**Slow Transaction Threshold**

- More than  % slower than the average of the last  hours
- Greater than  Milliseconds
- Greater than  3 Standard Deviations for the last  hours

**Very Slow Transaction Threshold**

- More than  % slower than the average of the last  hours
- Greater than  Milliseconds
- Greater than  4 Standard Deviations for the last  hours

**Stall Threshold**

- Disable Stall detection
- Stall occurs when a transaction takes more than  45 seconds.
- Stall occurs when a transaction's response time is  deviations above the average for the last 2 hours.

Apply to all Existing Business Transactions

For information about troubleshooting slow response times see [Troubleshoot Slow Response Times for PHP](#).

By default, errors with a severity of Error are detected from logged exceptions and messages. To view or modify error detection:

1. Click **Configure->Instrumentation** in the left navigation pane.
2. Click the Error Detection tab.
3. Click the PHP-Error Detection subtab.

See [Configure Error Detection for PHP](#) for information about configuring error detection. See [Trou](#)

[bleshoot Errors for PHP](#) for information about errors.