



APPDYNAMICS

# AppDynamics for PHP

AppDynamics Pro Documentation

Version 3.8.x

1. AppDynamics for PHP .....	3
1.1 Supported Environments and Versions for PHP .....	3
1.2 AppDynamics for PHP Architecture .....	8
1.3 Install the App Agent for PHP .....	8
1.3.1 Install the App Agent for PHP using a Shell Script .....	12
1.3.2 Install the App Agent for PHP using RPM .....	15
1.3.3 Upgrade the App Agent for PHP .....	18
1.3.4 Uninstall the App Agent for PHP .....	18
1.3.5 Running the PHP Proxy Daemon Manually .....	19
1.3.6 Resolve Installation Issues for PHP .....	20
1.4 Configure AppDynamics for PHP .....	21
1.4.1 Configure Transaction Detection for PHP .....	21
1.4.1.1 PHP Entry Points .....	24
1.4.1.1.1 Configure Web PHP Entry Points .....	24
1.4.1.1.2 Configure PHP MVC Entry Points .....	33
1.4.1.1.3 Configure PHP Drupal Entry Points .....	37
1.4.1.1.4 Configure PHP Wordpress Entry Points .....	40
1.4.1.1.5 Configure PHP Command Line Interface (CLI) Entry Points .....	43
1.4.1.1.6 Configure PHP Web Service Entry Points .....	47
1.4.1.2 Calibrate PHP Entry Point Detection .....	50
1.4.2 Configure Error Detection for PHP .....	57
1.4.3 Configure Call Graphs for PHP .....	62
1.5 Monitor PHP Applications .....	63
1.5.1 Distributed Transactions for PHP .....	63
1.5.2 Monitor PHP Backends .....	64
1.5.2.1 Monitor Predis Backends .....	64
1.5.2.2 Monitor RabbitMQ Backends for PHP .....	66
1.6 Troubleshoot PHP Application Problems .....	67
1.6.1 Troubleshoot Slow Response Times for PHP .....	69
1.6.2 Troubleshoot Errors for PHP .....	71
1.7 Tutorials for PHP .....	75
1.7.1 First Time Using the App Agent for PHP .....	75
1.7.2 Tutorial for PHP - Flow Maps .....	78
1.7.3 Tutorial for PHP - Server Health .....	83
1.7.4 Tutorial for PHP - Transaction Scorecards .....	86
1.8 Administer App Agents for PHP .....	89
1.8.1 App Agent for PHP Proxy Configuration Properties .....	89

# AppDynamics for PHP

This information covers using AppDynamics for PHP applications and environments. For general information see [AppDynamics Essentials](#) and [AppDynamics Features](#). For information about supported platforms see [Supported Platform Matrix for the PHP Agent](#).

---

## Tutorials

---

### Monitor PHP Application Problems

---

### Troubleshoot PHP Application Problems

---

### Install AppDynamics for PHP

---

### Configure AppDynamics for PHP

---

### Supported Environments and Versions for PHP

### AppDynamics for PHP Architecture

### Administer App Agents for PHP

## Supported Environments and Versions for PHP

- [Supported Platform Matrix for the App Agent for PHP](#)
  - [PHP Versions](#)
    - [PHP 5.2 Note](#)
    - [PHP ZTS Note](#)
  - [PHP Web Servers](#)
  - [Operating Systems](#)
  - [Architecture](#)
  - [PHP Frameworks and Protocols](#)
  - [Transaction Naming](#)
  - [HTTP Exit Points](#)
  - [Database Exit Points](#)
  - [Cache Exit Points](#)

- [Web Service Exit Points](#)
- [Message Queue Exit Points](#)
- [Opcode Cache Compatibility](#)
- [Correlation with AppDynamics for Databases](#)

## Supported Platform Matrix for the App Agent for PHP

### PHP Versions

Supported PHP Versions	Comment
5.2	Does not detect mysqli backends instantiated with the <b>new</b> keyword. See note below.
5.3	
5.4	
5.5	

#### PHP 5.2 Note

The app agent for PHP is incompatible with PHP 5.2 applications that use the **new** keyword to instantiate a mysqli backend.

For example, AppDynamics will not detect the mysqli backend created by a PHP 5.2 application that uses an expression like this:

```
// Does not get detected.  
$db = new mysqli("localhost", "user", "password", "database");
```

The workaround is to change such expressions to use `mysqli_connect()`:

```
$db = mysqli_connect("localhost", "user", "password", "database");
```

#### PHP ZTS Note

The app agent for PHP is incompatible with the mode of PHP called Zend Thread Safety (ZTS).

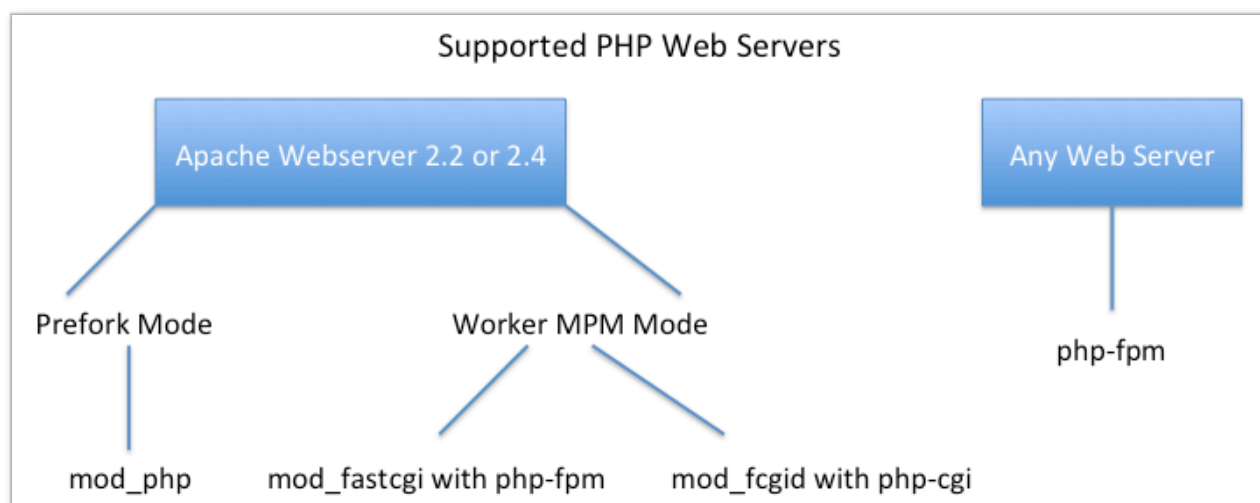
If you are using ZTS, AppDynamics suggests that you review your dependencies on ZTS to confirm that you actually need it, and if you do not, to switch to non-ZTS mode.

If you have a legacy infrastructure which requires ZTS or an app library that needs it, such as pthreads, contact AppDynamics Support.

### PHP Web Servers

Supported Web Server	Version	Comment
----------------------	---------	---------

Apache	2.2	in prefork mode using mod_php
Apache	2.4	in prefork mode using mod_php
Apache	2.2	in worker MPM mode using mod_fastcgi with php-fpm or mod_fcgid with php-cgi
Apache 2.4	2.4	in worker MPM mode using mod_fastcgi with php-fpm or mod_fcgid with php-cgi
Any Web Server compatible with php-fpm		



## Operating Systems

Supported Operating System	Version	Comment
RHEL/CentOS	5.8+	SELinux is disabled.
Ubuntu/Debian	12+	SELinux is disabled.

## Architecture

Supported Architecture
32-bit
64-bit

## PHP Frameworks and Protocols

Framework/Protocol	Version	Entry Point Type
Drupal	7	Drupal entry
WordPress	3.4 & 3.5	Wordpress
Zend	1 & 2	PHP MVC
CodeIgniter	2.x	PHP MVC
FuelPHP	1.5x & 1.6x	PHP MVC
Magento	1.5, 1.6 & 1.7	PHP MVC
Symfony	1 & 2	PHP MVC
CakePHP	2.x	PHP MVC
HTTP		PHP Web
CLI		PHP CLI

If your PHP framework is not listed here, the agent detects your entry points as PHP Web and names the business transactions based on the first two segments of the URI (the default naming convention for PHP Web transactions). So it is still possible to monitor applications on "unsupported" frameworks. You can modify the naming convention used for PHP Web Entry points. See [PHP Web Transaction Naming](#).

## Transaction Naming

Framework/Environment	Default Transaction Naming
Drupal	page callback name
Wordpress	template name
PHP MVC Frameworks	controller:action
PHP Modular MVC Frameworks	module:controller:action
PHP Web	URI
PHP Web Service <i>New in 3.8.2</i>	service name.operation name
PHP CLI	last two segments of the script's directory path plus the name of the script

Virtual host prefixing is available for all supported entry point types except PHP CLI.

## HTTP Exit Points

Supported HTTP Exit Points
curl/curl-multi
drupal_http_request()

fclose(), file_get_contents()
NuSOAP 0.9.5 <i>New in 3.8.2</i>
Zend_HTTP_Client::request()

## Database Exit Points

### Supported Database Exit Points

MySQL old native driver
MySQLi Extension
OCI8 <i>New in 3.8.2</i>
PDO

## Cache Exit Points

Supported Cache Exit Points	Version
-----------------------------	---------

Memcache	
Memcached	
Predis	0.8.5

Predis is supported on PHP versions 5.3 and higher.

Although Predis is a full PHP client library, the AppDynamics Agent for PHP supports Predis as an exit point only, not as an entry point.

## Web Service Exit Points

### Supported Web Service Exit Points

NuSOAP 0.9.5 <i>New in 3.8.2</i>
----------------------------------

## Message Queue Exit Points

### Supported Message Queue Exit Points

RabbitMQ
----------

RabbitMQ support requires the [amqp extension](#).

See also [Monitor RabbitMQ Backends for PHP](#).

## Opcode Cache Compatibility

Alternative PHP Cache (APC)
-----------------------------

## Correlation with AppDynamics for Databases

AppDynamics for Database version 2.7.4 or higher is required if you want to correlate the AppDynamics Agent for Database with the AppDynamics Agent for PHP.

## AppDynamics for PHP Architecture

The AppDynamics App Agent for PHP consists of:

- a PHP extension component
- a proxy component

The PHP extension component discovers, maps and tracks metrics for business transactions, app services, and backends in your web application by injecting instrumentation into the PHP application at runtime.

The proxy component is a Java daemon process that handles the communication between the PHP extension component and the Controller. The proxy reports the performance metrics to the Controller, where the data is stored, baselined, and analyzed. You can access this performance data interactively using the Controller console or programmatically using the AppDynamics REST API.

By default, the proxy component is automatically started when you start the PHP agent. Certain deployments require starting the proxy manually.



## Install the App Agent for PHP

- [Prerequisites for Agent Installation](#)
- [Install the Agent for PHP](#)
- [Files Added to Your Installation](#)
  - [.ini and .so files](#)
  - [Logs](#)
- [Special Procedures for PHP CLI](#)
- [Installing the Machine Agent on a PHP Node](#)
- [Post Installation Tasks](#)
- [Learn More](#)

You can install the app agent for PHP using either a Linux shell script (install.sh) or the RPM Package Manager (RPM).

These instructions assume that you are installing the AppDynamics App Agent for PHP in a



standard PHP environment, specifically:

- a single PHP installation running on the Linux machine
- PHP running in a single Apache or FPM pool
- standard packages have been used to install PHP, Apache and/or PHP-FPM
- no customizations have been made to your PHP configuration

It is possible that the installer will work if one or more of these assumptions is violated.

This installation results in an AppDynamics model of your application consisting of one application, one tier, and one node.

## Prerequisites for Agent Installation

1. Confirm that you have a [supported environment](#) installed on the server and that the server is configured correctly to run PHP scripts.

How you do this depends on your PHP environment. For example, in our Ubuntu 12+ web server running Apache we use:

```
sudo apt-get install apache2 php5 php5-cli  
php -i
```

In our CentOS 5+ web server running Apache mod\_ssl we use:

```
sudo yum install httpd mod_ssl php53 php53-cli  
php -i
```

2. Confirm that your PHP was not built with the **enable-debug** configure option.

The App Agent for PHP is incompatible with PHP builds that were compiled with debugging symbols.

To determine whether your PHP was built with debugging symbols you can use the following command:

```
php -i | grep -e "Debug Build"
```

The response should be:

```
Debug Build => no
```

3. Install the PHP application that you want to monitor, if it is not already installed.

4. Download the the appropriate PHP agent installer for your platform.

On RedHat and CentOS:

- To use the RPM installer for the 32-bit agent, download **appdynamics-php-agent-agent-version-number-i686.rpm**.

- To use the RPM installer for the 64-bit agent, download **appdynamics-php-agent-agent-version-number-x86\_64.rpm**.

On all other platforms:

- To use install.sh for the 32-bit agent, download **appdynamics-php-agent-x86-linux.tar.bz2**.
- To use install.sh for the 64-bit agent, download **appdynamics-php-agent-x64-linux.tar.bz2**.

If you use an on-premise Controller, download the latest version of the AppDynamics Controller. The download site is <http://download.appdynamics.com>.

5. Be prepared to provide the following information to the installation script:

- controller host and controller port: These are the host name or IP address and the port number of the AppDynamics controller that the agent connects to. SaaS customers receive this information from AppDynamics. On-premise customers establish these settings when they install the controller.
- AppDynamics application name: This is the name that you assign to the business application you will monitor with AppDynamics.
- AppDynamics tier name: This is the name that you assign to the tier you will monitor with AppDynamics.
- AppDynamics node name: This is the name of the basic unit of processing that the agent monitors.

If you have an on-premise AppDynamics controller running in multi-tenant mode or if you are using the AppDynamics SaaS Controller, you will also need to provide the following, which were included in your Welcome email from AppDynamics:

- AppDynamics account name
- AppDynamics account key

6. Stop the Apache server.

**Tip:** Do not install the AppDynamics PHP agent along with other non-AppDynamics Application Performance Management (APM) tools, especially in a production environment. The PHP agent installation may fail if there are other APM products installed in the same managed environment.

## Install the Agent for PHP

To use install.sh, see [Install the App Agent for PHP using a Shell Script](#).

To use RPM, see [Install the App Agent for PHP using RPM](#).

## Files Added to Your Installation

### .ini and .so files

The PHP agent installer adds the appdynamics\_agent.ini file to the directory that contains your php.ini file. You can find this directory using the following command:

```
php -i | grep -e "Additional .ini files parsed"
```

If the installer is not able to determine the directory where the ini fragments for your PHP deployment live, it displays the required AppDynamics ini fragment and prompts you to copy and paste it into your main php.ini file.

Also see <http://php.net/manual/en/configuration.file.php> for information about possible locations.

The installer also installs the appdynamics\_agent.so file in your PHP extensions directory. You can find this directory using the following command:

```
php -i | grep extension_dir
```

## Logs

There is an agent log and a proxy log for each application.

The agent log is located at \$<php\_agent\_install>/logs/agent.log. The log contains the transactions that the agent processes and then sends to the proxy. The default pattern for agent log naming is:

- agent.log: the current log
- agent.log.1: most recent log
- agent.log.2: second most recent log
- agent.log.3: third most recent log
- agent.log.4: fourth most recent log
- agent.log.5: fifth recent log

The proxy log is located \$<php\_agent\_install>/logs/proxy\_\$date.log. This log contains the transactions that the proxy accepts from the agent and then sends to the Controller.

## Special Procedures for PHP CLI

By default, the proxy component of the App Agent for PHP is configured to launch automatically when the agent starts up.

However, if you plan to instrument a PHP CLI entry point, you must arrange to run the proxy daemon manually.

If you have PHPs running CLI and apache on the same machine, your AppDynamics setup depends on whether you want all the traffic reported against a single AppDynamics node or separate nodes. A separate proxy is required for each AppDynamics node that you want to monitor in the controller.

If you want all the CLI traffic to be reported against one node and all the web traffic to be reported against a different node, configure apache to auto-launch the proxy (the default) and configure CLI to use a manually-launched proxy. This requires separate php.ini files - one for the web PHP with agent.auto\_launch\_proxy set to 1 and another for PHP CLI with agent.auto\_launch\_proxy set to 0.

If you want the web traffic and the CLI traffic to be reported against the same node, configure both apache and CLI to use the same manually launched proxy.

See [Running the PHP Proxy Daemon Manually](#) for details.

## Installing the Machine Agent on a PHP Node

If you install the machine agent on the machine hosting the instrumented PHP node and you specify the tier and node name in the machine agent's controller-info.xml file, the app agent for PHP will fail to register.

To avoid this problem:

- Install the app agent for PHP before you install the machine agent
- Do not specify the tier and node in the machine agent controller-info.xml, where it is optional. The machine agent will pick up the tier and node from the app agent configuration.

## Post Installation Tasks

Set any PHP proxy configuration properties that you have not already set in the install command.

See [App Agent for PHP Proxy Configuration Properties](#).

## Learn More

- [Install the App Agent for PHP using a Shell Script](#)
- [Install the App Agent for PHP using RPM](#)
- [Upgrade the App Agent for PHP](#)
- [Running the PHP Proxy Daemon Manually](#)
- [Resolve Installation Issues for PHP](#)
- [App Agent for PHP Proxy Configuration Properties](#)
- [Machine Agent Configuration Properties](#)
- [Install the Standalone Machine Agent](#)
- [AppDynamics for PHP Architecture](#)
- [Install the Controller](#)

## Install the App Agent for PHP using a Shell Script

- [Installing the App Agent for PHP](#)
  - [To install the app agent for PHP](#)
  - [Installation Samples](#)
- [Uninstall the Agent for PHP using install.sh](#)
- [Learn More](#)

Watch an AppDynamics engineer install the app agent for PHP in this AppDynamics in Action video:



## Installing the App Agent for PHP

To install the app agent for PHP

1. Untar the tarball containing the agent into a directory. The directory should be owned by the same user that runs Apache or php-fpm. AppDynamics recommends `/opt/appdynamics/php-agent`.

This documentation refers to this directory as the `php_agent_install` directory or `$<php_agent_install>`.

```
cd $<php_agent_install>
tar -xvzf appdynamics-php-agent-x64-linux.tar.bz2
```

2. Run the installation script using the following syntax. Each Apache installation is a single node.

```
$<php_agent_install>/install.sh [-s]
[-a=<account_name>@<account_key>]
[--http-proxy-host=<proxy_host>] [--http-proxy-port=<proxy_port>] [-e
<php_ext_dir>] [-i <php_ini_dir>]
<controller-host> <controller-port> <app_name> <tier_name>
<node_name>
```

- **-s option:** You can optionally specify the **-s** option if you want the agent to use SSL (HTTPS) to connect to the controller. In this case, set the Controller port to the HTTPS port of the controller.
- **account\_key and account name:** The AppDynamics `account_key` and `account_name` are required for a controller running in multi-tenant mode. These values are provided in your welcome email from AppDynamics.
- **http-proxy-host and http-proxy-port:** Set the `http-proxy-host` and `http-proxy-port` to route data to the controller through a proxy server. The `http-proxy-host` is the host name or IP address of the proxy server. The `http-proxy-port` is the proxy server's HTTP or HTTPS port, whichever you are using. If you set the `http-proxy-host` you must set the `http-proxy-port` as well. Proxy servers that require a username and password are not supported.
- **-e and -i options:** By default the installer uses the PHP CLI binary to determine where to install the app agent. This works for most PHP environments.

If you are using a different PHP, use the **-e** option to indicate the correct extensions directory for the appdynamics\_agent.so file and the **-i** option for the correct ini directory for the appdynamics\_agent.ini file.

3. Restart Apache, unless you are installing an agent to monitor PHP-CLI only.

If you are running multiple installations of Apache on the same machine, run install.sh once for each Apache, each time with the appropriate node, php\_ini dir and php\_ext dir options.

See [Files Added to Your Installation](#) for information about the default installation directories.

#### Installation Samples

Here is a sample command to install the agent on a single-tenant controller:

```
install.sh controller 8090 myApp myTier myNode
```

Here is a sample command to install the agent using SSL on a multi-tenant on-premise controller:

```
install.sh -s -a=PHPCust1000@9456d222-66e2-54d2-f8aabbcb66c4e  
controller1.appdynamics.com 8818 myApp myTier myNode
```

Here is a sample command to route traffic to the controller through a proxy server.

```
install.sh --http-proxy-host=myproxyhost --http-proxy-port=8099  
controller 8090 myApp myTier myNode
```

**Tip:** The installer overwrites your existing settings in the controller-info.xml file. If you configured properties in that file, you need to update them every time you run the installer.

If the startup does not succeed, [file a support ticket](#).

### Uninstall the Agent for PHP using install.sh

If you installed the agent using install.sh, use install.sh to uninstall it.

To uninstall:

1. Shut down Apache.
2. From the PHP agent install directory, run the PHP installer with the -u option:

```
install.sh -u
```

3. Delete the <php\_agent\_install> directory.

### Learn More

- [Install the App Agent for PHP](#)
- [Install the App Agent for PHP using RPM](#)
- [Upgrade the App Agent for PHP](#)
- [Running the PHP Proxy Daemon Manually](#)
- [App Agent for PHP Proxy Configuration Properties](#)
- [AppDynamics for PHP Architecture](#)
- [Install the Controller](#)

## Install the App Agent for PHP using RPM

- [Procedure for Installing the Agent for PHP Using RPM](#)
- [RPM Environment Variables](#)
- [Run the RPM Package](#)
  - [Using sudo to install](#)
  - [Updating the Installation](#)
- [RPM Log File](#)
- [Uninstall the Agent for PHP using RPM](#)
- [Learn More](#)

The RPM package lets you automate the installation of the app agent for PHP.

RPM is supported on RHEL and CentOS. You must run the package as root.

RPM installs one agent at a time. Installation of multiple agents is not supported.

### Procedure for Installing the Agent for PHP Using RPM

1. Download the RPM package from the [AppDynamics download site](#).
2. Set the environment variables. RPM gets its installation information from the environment, not from the command-line. See [RPM Environment Variables](#).
3. Run the RPM package. See [Run the RPM Package](#).
4. If there are errors, examine the log file. See [RPM Log File](#).
5. Restart Apache, unless you are installing an agent to monitor PHP CLI only.

### RPM Environment Variables

The RPM installer attempts to determine the location of your PHP installation based on the PATH environment variable. It uses the first PHP installation that it encounters in the PATH to configure the installer. If you have installed PHP in a non-standard location, you must provide the directory of your PHP binary in APPD\_PHP\_PATH.

You can route data to the controller through a proxy server, but proxy servers that require a username and password are not supported.

The installer uses the default values for the other variables if you do not set them.

Set the APPD environment variables at the operating system level. You may want to use a script to set the environment variables.

Environment Variable	Description	Default	Required?
----------------------	-------------	---------	-----------

APPD_PHP_PATH	Directory containing the PHP binary	None	if your PHP binary is not in a standard location. By default the installer uses the PHP CLI binary to determine where to install the app agent.
APPD_CONF_CONTROLLER_HOST	Controller host name	"localhost"	
APPD_CONF_CONTROLLER_PORT	Controller port	8080	
APPD_CONF_APP	Application name	"MyApp"	
APPD_CONF_TIER	Tier name	Hostname of the machine running the script (same as the node name)	
APPD_CONF_NODE	Node name	Hostname of the machine running the script	
APPD_CONF_ACCOUNT_NAME	Account name	None	if you have an on-premise AppDynamics controller running in multi-tenant mode or if you are using the AppDynamics SaaS Controller.
APPD_CONF_ACCOUNT_KEY	Account key	None	if you have an on-premise AppDynamics controller running in multi-tenant mode or if you are using the AppDynamics SaaS Controller.
APPD_CONF_SSL_ENABLED	true to enable SSL communication with the controller, false otherwise	false	
APPD_CONF_HTTP_PROXY_HOST	Hostname or IP address of the http proxy server	None	if you want to route data to the controller through a proxy server.



APPD_CONF_HTTP_PROXY_PORT	HTTP or HTTPS port of the http proxy server; must be set if APPD_CONF_HTTP_PROXY_HOST is set	None	if you want to route data to the controller through a proxy server.
---------------------------	--	------	---

## Run the RPM Package

To run the installer package:

```
rpm -i <package-name>
```

If you have multiple installations of PHP on one machine, run the package once for each PHP installation, each time with the appropriate APPD\_PHP\_PATH and APPD\_CONF\_NODE settings.

### Using sudo to install

If you are using sudo to pass the environment variables to the installation script you can use:

```
sudo APPD_PHP_PATH=/opt/php rpm -i <package-name>
```

or

```
APPD_PHP_PATH=/opt/php sudo -E rpm -i <package-name>
```

## Updating the Installation

Any changes that you made to the configuration files are preserved when you re-run the installer. RPM saves your original settings and appdynamics\_agent\_log4cxx.xml files with the settings from the previous installation.

## RPM Log File

If the installation succeeds, no log file is generated.

If there were errors, a message displays the location of the log file generated in the /tmp directory. Examine this log file to identify the cause of the problem.

## Uninstall the Agent for PHP using RPM

If you installed the agent using RPM, use RPM to uninstall it.  
To uninstall:

```
rpm -e appdynamics-php-agent-<version>
```

The existing configurations are saved in a tarball in /tmp, the location of which will be displayed

after the uninstall completes.

### Learn More

- [Install the App Agent for PHP](#)
- [Install the App Agent for PHP using a Shell Script](#)
- [Running the PHP Proxy Daemon Manually](#)
- [Resolve Installation Issues for PHP](#)
- [App Agent for PHP Proxy Configuration Properties](#)
- [AppDynamics for PHP Architecture](#)
- [Install the Controller](#)


### Upgrade the App Agent for PHP

- [To upgrade the app agent](#)

[Learn More](#)

To upgrade the app agent

1. Shut down the web server or php-fpm.
2. Copy the controller host, controller port, application name, tier name and node name property values from your <php\_agent\_install>/proxy/conf/controller-info.xml file.  
If you are running in multi-tenant mode, also copy the account name and account access key property values.
3. Recursively remove or rename the old AppDynamics PHP installation directory.
4. Download and extract the most recent agent tarball.
5. Run the installation script, using the values that you copied from controller-info.xml for the parameters.
6. Restart Apache or php-fpm.

 If you are using the agent to monitor PHP CLI without running a web server, you can omit steps 1 and 6.

### Learn More


- [Install the App Agent for PHP](#)
- [Uninstall the App Agent for PHP](#)
- [Resolve Installation Issues for PHP](#)
- [App Agent for PHP Proxy Configuration Properties](#)

### Uninstall the App Agent for PHP

To uninstall the app agent

1. Shut down Apache.
2. Delete the appdynamics\_agent.ini file .
3. Delete the appdynamics\_agent.so file.

4. Delete the <php\_agent\_install> directory.

 If you are using the agent to monitor PHP CLI without running a web server, you can omit step 1.

See [Install the App Agent for PHP](#) for suggestions on how to locate the .ini and .so files.

## Running the PHP Proxy Daemon Manually

- [Configuring Startup of the runproxy Script](#)
  - [To configure manual starting of runproxy](#)
- [Executing the runproxy Script](#)

By default, when the PHP agent starts up, it automatically executes the runproxy shell script. This script runs the Java proxy daemon that handles communication between the PHP agent and the controller. See [AppDynamics for PHP Architecture](#) for information about how the Java proxy daemon fits into the PHP agent architecture.

Automatic startup of the proxy works for the great majority of situations. However, you can suppress the automatic startup of this script and run it manually. You would do this if:

- You plan to instrument a PHP CLI entry point. The PHP CLI entry point requires manual startup of the proxy.
- You need to set the Java system properties that the proxy reads.
- You need to change the JRE that the proxy uses or to change the maximum heap size.
- You need to change the maximum heap size.

## Configuring Startup of the runproxy Script

You need to configure the agent for manual startup if you plan to start the runproxy script manually.

### To configure manual starting of runproxy


1. By default the agent.auto\_launch\_proxy setting in php.ini or appdynamics\_agent.ini is set to 1 to enable automatic startup of the proxy.

Change it to 0 to suppress automatic startup if you want to execute runProxy manually.

2. Edit the runproxy script as needed. The runproxy script is in the <php\_agent\_install>/proxy directory.

3. If you edited the runproxy script, save your version in the <php\_agent\_install>/proxy directory. Save your version in a different file instead of overwriting the original script. That way you can easily switch between the default and alternate scripts.

4. In php.ini, set the agent.proxy\_script to the path of the runproxy that you want to use. The file in the script is relative to the root of the PHP agent. You can specify the absolute path if you prefer.

 Whenever you install the PHP agent, the installer overwrites the runproxy script and the appdynamics\_agent.ini file. If you re-install, you need to reset the agent.auto\_launch\_proxy setting in the appdynamics\_agent.ini file before you restart the server.

## Executing the runproxy Script

Before any traffic is run on the instrumented server, execute the appropriate version of the runproxy script in the <php\_agent\_install>/proxy directory to start the proxy.

Every time you reboot the server, you need to execute the runproxy script if you have opted to start the proxy manually.

## Resolve Installation Issues for PHP

If you installed the agent, started up your instrumented server, and your application is receiving traffic but no metrics are being reported, try these suggestions for investigating installation issues.

### Determine whether the installer installed in the correct directory.

It is possible that the agent was installed in the wrong directory. Verify the location of your PHP installation.

- Verify the location of your PHP by running phpinfo. See <http://us1.php.net/phpinfo>.

Then check where the installer actually installed the agent files.

- The appdynamics\_agent.ini file should be in the same directory that contains the php.ini file for your PHP installation.
- The appdynamics\_agent.so file should be in the extensions directory for your PHP installation.

See [Files Added to Your Installation](#) for information about how to locate these directories.

In addition, on Linux you can use pstree to locate the agent. Pstree displays the AppDynamics agent running under Apache if the agent is installed properly. See <http://freecode.com/projects/pstree>.

If the agent files are not in the correct directories, re-install the agent with the **-i** and **-e** options.

#### To reinstall the agent

If app agent is not installed in the right directory, re-install the agent using the install.sh installer with the **-i** and **-e** options. Use the **-i** to install the appdynamics\_agent.ini file in the same directory as your php.ini file and the **-e** to install the appdynamics\_agent.so file in the same directory as your PHP extensions directory. See [Install the App Agent for PHP using a Shell Script](#).

If you initially installed the agent using the RPM installer, you can find the shell script installer at /usr/lib/appdynamics-php5/install.sh.

### Confirm that the proxy is running.

The Java proxy is the part of the agent that communicates with the Controller. If the agent is installed in the right place, confirm that the Java proxy is running.

#### To confirm whether the proxy is running

1. From the command line enter 'ps aux | grep java'.

2. Inspect the list. You should see output similar to the following if the proxy is running:

```
/usr/lib/appdynamics-php5/proxy/jre/bin/java -server -Xmx120m
-classpath
/usr/lib/appdynamics-php5/proxy/conf/logging/*:/usr/lib/appdynamics-
php5/proxy/lib/*:/usr/lib/appdynamics-php5/proxy/lib/tp/*:/usr/lib/a
ppdynamics-php5/proxy/*
-Djava.library.path=/usr/lib/appdynamics-php5/proxy/lib/tp
-Dappdynamics.agent.logs.dir=/usr/lib/appdynamics-php5/logs
-Dcomm=/tmp/ad-siJ4rp -DagentType=PHP_APP_AGENT
-Dappdynamics.agent.runtime.dir=/usr/lib/appdynamics-php5/proxy
com.appdynamics.ee.agent.proxy.kernel.Proxy
```

If you are instrumenting a PHP CLI script, you need to start the proxy manually. You may also need to start the proxy manually if you have special requirements for running Java processes. See [Running the PHP Proxy Daemon Manually](#) .

## Check the Proxy Configuration Properties

It is possible that the properties that the proxy uses to communicate with the Controller were not set properly.

You can modify these properties in the controller-info.xml file located in the <php\_agent\_install>/proxy/conf directory.

See [App Agent for PHP Proxy Configuration Properties](#) for details about these properties.

## Configure AppDynamics for PHP

### Configure Transaction Detection for PHP

- [Accessing Transaction Detection](#)
  - [To Access Business Transaction Detection Configuration](#)
- [PHP Entry Points](#)
- [Business Transaction Naming and Identification](#)
  - [Virtual Host Naming Prefix](#)
- [Learn More](#)

### Accessing Transaction Detection

**To Access Business Transaction Detection Configuration**

1. From the left navigation pane select **Configure -> Instrumentation**.
2. Click the Transaction Detection tab if it is not already selected.

3. Click the PHP - Transaction Detection tab.

4. Do one of the following:

- To configure transaction detection at the application level, in the left panel select the application.  
If you select the application, you can optionally click the button to configure all tiers to use the application-level configuration.
- To configure transaction detection at the tier level, in the left panel select the tier for which you want to configure transaction detection.  
You can choose the button to apply the application configuration to the selected tier or the button to create a custom configuration for this tier.

5. To configure a custom configuration for the tier, select **Use Custom Configuration for this Tier**.

## PHP Entry Points

The entry point is where the business transaction begins. Typically an entry point is a method or operation.

The following screenshot shows PHP entry point types for the transaction types that AppDynamics automatically detects and monitors. The entry point types are based on the PHP framework.







For each type, you can enable and disable transaction monitoring. When monitoring is disabled, the agent stops counting, measuring, recording, etc. all activity on servers of that type.

If the agent cannot detect the framework, the entry point type defaults to PHP Web. This entry point automatically detects all HTTP requests to the application.

Java - Transaction Detection
.NET - Transaction Detection
**PHP - Transaction Detection**

Copy
Configure all Tiers to use this Configuration

▼ Entry Points

Type	Transaction Monitoring	Automatic Transaction Detection
 PHP Web	<input checked="" type="checkbox"/> Enabled	<a href="#">Configure Naming</a> <input type="checkbox"/> Use Virtual Host in Business Transaction names
 PHP CLI	<input checked="" type="checkbox"/> Enabled	
 PHP MVC	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Use Virtual Host in Business Transaction names
 Drupal	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Use Virtual Host in Business Transaction names
 Wordpress	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Use Virtual Host in Business Transaction names
 PHP Web Service	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Use Virtual Host in Business Transaction names

## Business Transaction Naming and Identification

AppDynamics identifies and names business transactions using the following conventions:

- **PHP Web:** the first two segments of the URI
- **PHP MVC:**
  - <module> : <controller> : <action> for modular frameworks
  - <controller> : <action> for non-modular frameworks
- **PHP CLI:** last two segments of the script's directory path plus the name of the script
- **PHP Web Service:** <ServiceName>.<OperationName>
- **Drupal:** <page callback name>
- **Wordpress:** <page template>

### Virtual Host Naming Prefix

If you have multiple virtual hosts configured on a single web server, you can differentiate among the business transactions by checking the Use Virtual Host in Business Transaction names check box. This adds the virtual host name as a prefix to the default business transaction name for all

transactions subsequent to this configuration.

For example, an application has two exit.php actions running on the same physical host. If the virtual host name is not used to name the transaction, the AppDynamics agent identifies all requests to exit.php as a single business transaction named "exit.php". However, if one of those actions runs on a virtual host named "phpagent1" and the other runs on a virtual host named "phpagent2" and the virtual host name is used to name the transaction, the agent identifies two different business transactions named "phpagent1 : exit.php" and "phpagent2 : exit.php".

## Learn More

- [Business Transaction Monitoring](#)
- [PHP Entry Points](#)

## PHP Entry Points

The entry point is where the business transaction begins, typically a URI or a page callback name, MVC controller action, page template name, or page template.

### Configure Web PHP Entry Points

- [PHP Web Transaction Naming](#)
  - [Transaction Naming Based on Web Context](#)
    - [Web Context is http://example.com/store/checkout](#)
    - [Web Context is http://example.com/secure/internal/updateinventory](#)
    - [Web Context is http://example.com/secure/orders/process.creditcard](#)
    - [To configure PHP Web Transaction Naming](#)
- [Default Detection for PHP Web Entry Points](#)
- [Configure Custom Match Rules for PHP Web Entry Points](#)
  - [To configure a custom match rule for a PHP Web entry point](#)
  - [To modify a custom match rule for a PHP Web entry point](#)
  - [To remove a custom match rule for a PHP Web entry point](#)
- [Configure Exclude Rules for PHP Web Entry Points](#)
  - [To configure an exclude rule for a PHP Web entry point](#)
  - [To modify an exclude rule for a PHP Web entry point](#)
  - [To remove an exclude rule for a PHP Web entry point](#)
- [Split PHP Web Transactions](#)
  - [To split transactions for a PHP Web entry point using a custom match rule](#)
- [Learn More](#)

If the app agent for PHP cannot detect the application's framework, the entry point type defaults to PHP Web. This entry point automatically detects all HTTP requests to the application and names the business transaction after the URI.

### PHP Web Transaction Naming

The default convention for PHP Web transactions is to use first two segments of the URI to name the transaction. You can change the naming convention to use the full URI or to use different segments of the URI.

By default, AppDynamics automatically names PHP Web transactions based on the URI of the application.

You can also configure dynamic transaction naming based on the details of the user request.



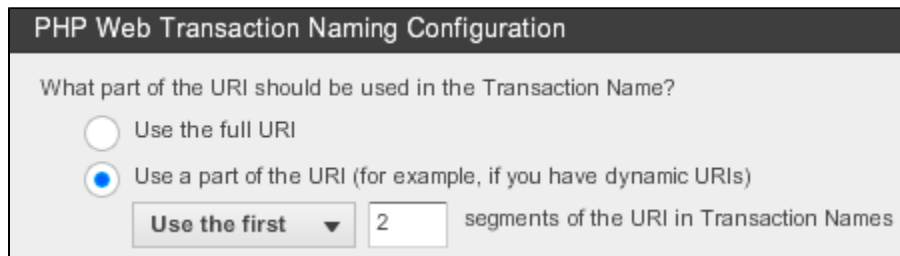
### ***Transaction Naming Based on Web Context***

In certain situations, it is appropriate to modify the existing naming and name business transactions based on different web contexts.

For example, MyOnlineBiz's entry point tier has multiple contexts deployed in a single PHP web server instance. These web contexts represent different parts of the same business application and therefore require following different naming strategies. Some examples:

Web Context is <http://example.com/store/checkout>

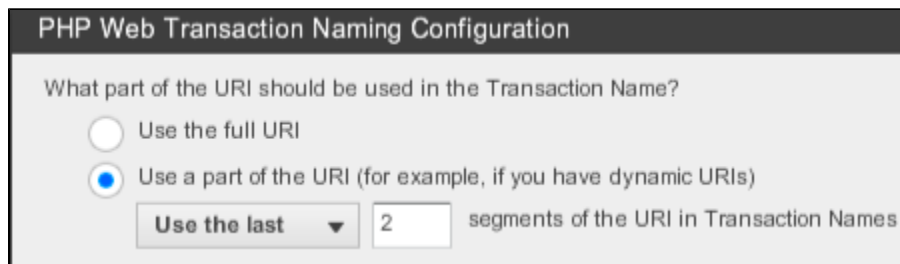
The naming strategy should use first two segments to name these transactions as: **/store/checkout**.



The screenshot shows a configuration window titled "PHP Web Transaction Naming Configuration". It asks "What part of the URI should be used in the Transaction Name?". There are two radio button options: "Use the full URI" (unselected) and "Use a part of the URI (for example, if you have dynamic URIs)" (selected). Below the radio buttons, there is a dropdown menu set to "Use the first" and a text input field containing the number "2", followed by the text "segments of the URI in Transaction Names".

Web Context is <http://example.com/secure/internal/updateinventory>

The naming strategy should use last two segments to name these requests as: **/internal/updateinventory**.



The screenshot shows a configuration window titled "PHP Web Transaction Naming Configuration". It asks "What part of the URI should be used in the Transaction Name?". There are two radio button options: "Use the full URI" (unselected) and "Use a part of the URI (for example, if you have dynamic URIs)" (selected). Below the radio buttons, there is a dropdown menu set to "Use the last" and a text input field containing the number "2", followed by the text "segments of the URI in Transaction Names".

Web Context is <http://example.com/secure/orders/process.creditcard>

The naming strategy should use the combination of parameter value for "type" and the last two segments to name such requests as: **/orders/process.creditcard**.

**PHP Web Transaction Naming Configuration**

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

☐ Use the full URI  
☒ Use a part of the URI (for example, if you have dynamic URIs)

Use the last 2 segments of the URI in Transaction Name

☒ Name Transactions dynamically using part of the request

☐ Use URI segment(s) in Transaction names  
 Segment Numbers  *Enter a comma separated list of parameters*

☒ Use a parameter value in Transaction names  
 Parameter Name type



To configure PHP Web Transaction Naming

1. Click **Configure Naming** in the row for the PHP Entry point in the Entry Points table.

Java - Transaction Detection   .NET - Transaction Detection   **PHP - Transaction Detection**

Copy   Configure all Tiers to use this Configuration

▼ Entry Points

Type	Transaction Monitoring	Automatic Transaction Detection
 PHP Web	<input checked="" type="checkbox"/> Enabled	<a href="#">Configure Naming</a>  <input type="checkbox"/> Use Virtual Host in Business Transaction names

2. In the naming configuration screen do one of the following:

- To use the full URI as the transaction name select **Use the full URI**.

or

- To use specify segments to use as the transaction name, select **Use the first** or **Use the last** from the dropdown menu and then enter the number of segments to use in the text field.

The screenshot shows a dialog box titled "PHP Web Transaction Naming Configuration". It contains the following elements:

- A question: "What part of the URI should be used in the Transaction Name?"
- Two radio buttons:
  - ☐ Use the full URI
  - ☒ Use a part of the URI (for example, if you have dynamic URIs)
- A dropdown menu set to "Use the first" and a text input field containing "2", followed by the text "segments of the URI in Transaction Names" and a link "What does this do?".
- A checked checkbox: "Name Transactions dynamically using part of the request"
- Five radio buttons for dynamic naming options:
  - ☐ Use URI segment(s) in Transaction names. Below it is a text input field labeled "Segment Numbers" with the placeholder text "Enter a comma separated list of parameter numbers (e.g. 1,3,4)".
  - ☒ Use a parameter value in Transaction names. Below it is a text input field labeled "Parameter Name" containing the value "item".
  - ☐ Use a header value in Transaction names. Below it is a text input field labeled "Header Name".
  - ☐ Use a cookie value in Transaction names. Below it is a text input field labeled "Cookie Name".
  - ☐ Use the request method (GET/POST/PUT) in Transaction names.
- At the bottom right are "Cancel" and "Save" buttons.

### 3. To configure dynamic naming:

- Check the **Name Transactions dynamically using part of the request** check box.
- Select the radio button that corresponds to the part of the request that you want to use in the transaction name.
- Enter the request value in the text field where appropriate.

### 4. Click **Save**.

#### Default Detection for PHP Web Entry Points

By default, AppDynamics automatically names PHP Web transactions based on the url of the application.

You may be seeing too many or too few PHP Web transactions. If you are not getting the visibility that you need with the default configuration, create one or more custom match rules for PHP Web entry points.

If you are seeing too many transactions, create exclude rules to prevent the agent from discovering the entry points that you do not need to monitor. Or create custom match rules that group several entry points into a single entry point.

If you are seeing too few transactions, review your custom match rules rules to make sure that they detect all the entry points that you want the agent to discover.

#### Configure Custom Match Rules for PHP Web Entry Points

You can configure multiple match criteria for the agent to use to detect the transaction.

The agent determines the match using only the configured criteria.

To configure a custom match rule for a PHP Web entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. Click the add icon to add a new custom match rule.
4. Select **PHP Web** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the New Business Transaction Match Rule-PHP Web window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. Optionally enter the priority of this rule for the app agent to use when multiple rules could apply to the same entry point. See [The Priority Parameter when Multiple Rules Apply](#).
9. In the Transaction Match Criteria tab, for each match conditions that you want to specify, check the check box for the condition for which you are configuring a match. Then do one or more of the following:
  - **HTTP method:** To configure a match on the HTTP method, select the method (GET, POST, PUT, DELETE) from the dropdown list.

- **URI:** To configure a match on the URI, enter the value against which to match the in the text field and the the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.
- **HTTP parameter:** To configure a match on an HTTP parameter, from the dropdown list select **Check for parameter existence** or **Check for parameter value** and enter the parameter name or the parameter value match condition as appropriate.  
To configure more than one parameter, click the + icon to get a new HTTP parameter panel.
- **Header:** To configure a match on the header, from the dropdown list select **Check for parameter existence** or **Check for parameter value** and enter the parameter name or the parameter value match condition as appropriate.  
To configure more than one parameter, click the + icon to get a new header panel.
- **Port:** To configure a match on the port, enter the value against which to match the in the text field and the the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.
- **Cookie:** To configure a match on a cookie, from the dropdown list select **Check for cookie existence** or **Check for cookie value** and enter the cookie name or the cookie value match condition as appropriate.  
To configure more than one cookie, click the + icon to get a new cookie panel.

## 9. Click **Create Custom Match Rule**.

To modify a custom match rule for a PHP Web entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 6 through 9 in [To configure a custom match rule for a PHP Web entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

To remove a custom match rule for a PHP Web entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

## Configure Exclude Rules for PHP Web Entry Points

You can configure an exclude rule to exclude from detection any entry point that matches the

configured match conditions. The match conditions are the same as those available for [configuring custom match rules](#).

Transactions excluded through exclude rules are not counted towards the business transaction limit of 50 transactions per agent or 200 transactions per application.

To configure an exclude rule for a PHP Web entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. Click the add icon to add a new exclude rule.
4. Select **PHP Web** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the New Exclude Business Transaction Match Rule-PHP Web window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. In the Transaction Match Criteria tab, for each match conditions that you want to specify, check the check box for the condition for which you are configuring a match. Then configure one or more of the following:
  - **HTTP method:** To configure a match on the HTTP method, select the method (GET, POST, PUT, DELETE) from the dropdown list.
  - **URI:** To configure a match on the URI, enter the value against which to match the in the text field and the the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.
  - **HTTP parameter:** To configure a match on an HTTP parameter, from the dropdown list select **Check for parameter existence** or **Check for parameter value** and enter the parameter name or the parameter value match condition as appropriate.  
To configure more than one parameter, click the + icon to get a new HTTP parameter panel.
  - **Header:** To configure a match on the header, from the dropdown list select **Check for parameter existence** or **Check for parameter value** and enter the parameter name or the parameter value match condition as appropriate.  
To configure more than one parameter, click the + icon to get a new header panel.
  - **Port:** To configure a match on the port, enter the value against which to match the in the text field and the the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.
  - **Cookie:** To configure a match on a cookie, from the dropdown list select **Check for cookie existence** or **Check for cookie value** and enter the cookie name or the cookie value match condition as appropriate.  
To configure more than one cookie, click the + icon to get a new cookie panel.
9. Click **Create Exclude Rule**.

New Exclude Business Transaction Match Rule - PHP Web

Name

Enabled ☒

**Transaction Match Criteria** | Split Transactions Using Request Data

☐ Method

☐ URI

☐ HTTP Parameter (Both GET query parameters and POST parameters can be used)

Parameter Name

Value

☐ Header

Parameter Name

Value

☐ Port

☐ Cookie

Cookie Name

Cancel Create Exclude Rule

To modify an exclude rule for a PHP Web entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 and 8 in [To configure an exclude rule for a PHP Web entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

To remove an exclude rule for a PHP Web entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

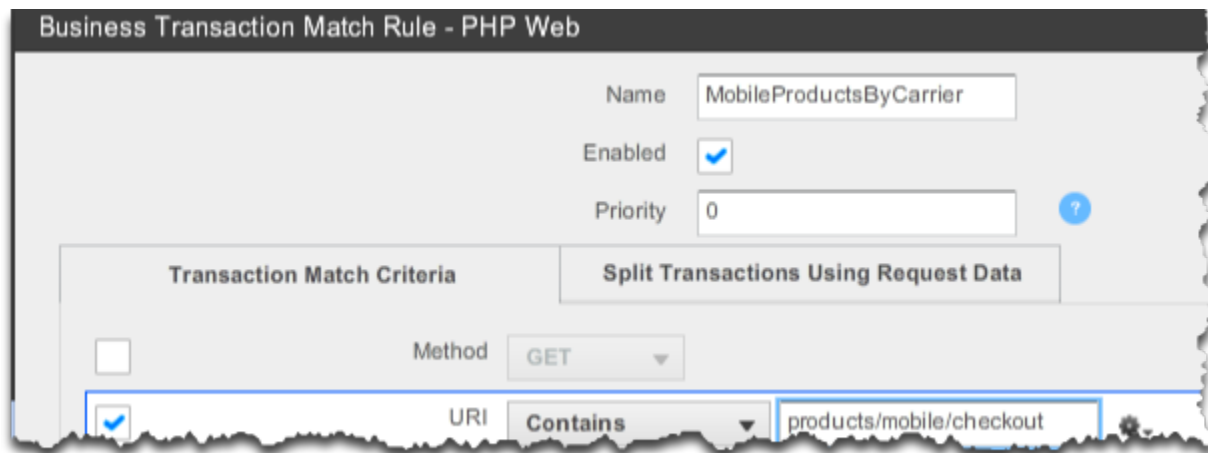
#### Split PHP Web Transactions

If a transaction is configured to identify the entry point based on the the URI, you can optionally

split the transaction into multiple transactions. For example, a login request may be automatically detected as a single transaction, but you want to split it into two transactions based on whether the request branches to a new-user or existing-user operation. For general information about transaction splitting see [Transaction Splitting for Dynamic Discovery](#).

To split transactions for a PHP Web entry point using a custom match rule

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule for which you want to split the transaction.
4. Double-click the rule or click the edit icon.
5. Verify in the **Transaction Match Criteria** tab that the rule is configured to identify the entry point based on the URI.



6. Click the **Split Transaction Using Request Data** tab.
7. Check the Split Transactions using request data check box.
8. Select the part of the URI that you want to use to split the transaction.  
In the example below the transaction is split by the carrier parameter. As a result, separate transactions will be detected for
  - products/mobile/checkout?carrier=verizon
  - products/mobile/checkout?carrier=sprint
  - products/mobile/checkout?carrier=att



**Business Transaction Match Rule - PHP Web**

Name

Enabled ☒

Priority  ?

**Transaction Match Criteria** | **Split Transactions Using Request Data**

☒ Split Transactions using request data ?

☐ Use the first  segments in Transaction names

☐ Use the last  segments in Transaction names

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

☒ Use a parameter value in Transaction names  
Parameter Name

☐ Use a header value in Transaction names  
Header Name

☐ Use a cookie value in Transaction names  
Cookie Name

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

9. Click **Save**.

Learn More

- [Configure Transaction Detection for PHP](#)
- [Organizing Traffic as Business Transactions](#)
- [Match Rule Conditions](#)
- [Regular Expressions In Match Conditions](#)

#### Configure PHP MVC Entry Points

- [Default Detection and Naming for PHP MVC Entry Points](#)
- [Configure Custom Match Rules for PHP MVC Entry Points](#)
  - [To configure a custom match rule for a PHP MVC entry point](#)
  - [To modify a custom match rule for a PHP MVC entry point](#)
  - [To remove a custom match rule for a PHP MVC entry point](#)
- [Configure Exclude Rules for PHP MVC Entry Points](#)
  - [To configure an exclude rule for a PHP MVC entry point](#)
  - [To modify an exclude rule for a PHP MVC entry point](#)
  - [To remove an exclude rule for a PHP MVC entry point](#)
- [Learn More](#)

By default, automatic transaction detection for PHP MVC transactions is turned off. See [Configure Transaction Detection for PHP](#) for general information about turning on automatic detection in the

PHP entry point window.

#### Default Detection and Naming for PHP MVC Entry Points

By default, for most MVC frameworks, the business transaction is named using the controller:action. For modular MVC frameworks, the business transaction is named using the module:controller:action.

You may be seeing too many or too few PHP MVC transactions. If you are not getting the visibility that you need with the default configuration, create one or more custom match rules for PHP MVC entry points.

If you are seeing too many transactions, create exclude rules to prevent the agent from discovering the entry points that you do not need to monitor. Or create custom match rules that group several entry points into a single entry point.

If you are seeing too few transactions, review your custom match rules rules to make sure that they detect all the entry points that you want the agent to discover.

#### Configure Custom Match Rules for PHP MVC Entry Points

You can configure a custom match rule that matches a portion of the module, controller or view name (such as "Begins with" or "Contains") rather than the default which matches the entire name ("Equals").

You cannot configure a single match rule that matches on both the controller and the action.

The agent determines the match using only the configured criteria. So if you configure a rule based on matching a specific controller pattern but do not configure the action or the module, the agent does not consider the action or the module in evaluating the match.

To configure a custom match rule for a PHP MVC entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. Click the add icon to add a new custom match rule.
4. Select **PHP MVC** from the Entry Point Type dropdown list.
5. Click **Next**.

6. In the New Business Transaction Match Rule-PHP MVC window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. In the Business Transaction Match Criteria section, for each match condition that you want to specify, do the following:
  - a. Check the check box for the segment for which you are configuring a match.
  - b. In the text field, enter the value against which to match.
  - c. Select the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.
9. Click **Create Custom Match Rule**.

**To modify a custom match rule for a PHP MVC entry point**

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 through 10 in [To configure a custom match rule for a PHP MVC entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

**To remove a custom match rule for a PHP MVC entry point**

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

**Configure Exclude Rules for PHP MVC Entry Points**

You can configure an exclude rule to exclude from detection any entry point that matches the configured match conditions. The match conditions are the same as those available for [configuring a custom match rule](#).

Transactions excluded through exclude rules are not counted towards the business transaction limit of 50 transactions per agent or 200 transactions per application.

**To configure an exclude rule for a PHP MVC entry point**

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.

3. Click the add icon to add a new exclude rule.
4. Select **PHP MVC** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the Exclude Business Transaction Match Rule-PHP MVC window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. In the Business Transaction Match Criteria section, for each match condition that you want to specify, do the following:
  - a. Check the check box for the segment for which you are configuring a match.
  - b. In the text field enter the value against which to match.
  - c. Select the condition to use to determine the match from the dropdown list.
To reverse the condition, click the gear icon and check the NOT condition check box.
9. Click **Create Exclude Rule**.

To modify an exclude rule for a PHP MVC entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 and 8 in [To configure an exclude rule for a PHP MVC entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

To remove an exclude rule for a PHP MVC entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to remove.

4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

#### Learn More

- [Configure Transaction Detection for PHP](#)
- [Organizing Traffic as Business Transactions](#)
- [Match Rule Conditions](#)
- [Regular Expressions In Match Conditions](#)

#### Configure PHP Drupal Entry Points

- [Default Detection and Naming for Drupal Entry Points](#)
- [Configure Custom Match Rules for Drupal Entry Points](#)
  - [To configure a custom match rule for a Drupal entry point](#)
  - [To modify a custom match rule for a Drupal entry point](#)
  - [To remove a custom match rule for a Drupal entry point](#)
- [Configure Exclude Rules for Drupal Entry Points](#)
  - [To configure an exclude rule for a Drupal entry point](#)
  - [To modify an exclude rule for a Drupal entry point](#)
  - [To remove an exclude rule for a Drupal entry point](#)
- [Learn More](#)

By default, automatic transaction detection for Drupal transactions is turned off. See [Configure Transaction Detection for PHP](#) for general information about turning on automatic detection in the PHP entry point window.

#### Default Detection and Naming for Drupal Entry Points

By default, AppDynamics automatically names Drupal transactions based on the page callback name of the Drupal module.

You may be seeing too many or too few Drupal transactions. If you are not getting the visibility that you need with the default configuration, create one or more custom match rules for Drupal entry points.

If you are seeing too many transactions, create exclude rules to prevent the agent from discovering the entry points that you do not need to monitor. Or create custom match rules that group several entry points into a single entry point.

If you are seeing too few transactions, review your custom match rules rules to make sure that they detect all the entry points that you want the agent to discover.

#### Configure Custom Match Rules for Drupal Entry Points

You can configure a custom match rule that matches a portion of the page callback name (such as "Begins with" or "Contains") rather than the default which matches the entire page callback name ("Equals").

To configure a custom match rule for a Drupal entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. Click the add icon to add a new custom match rule.
4. Select **Drupal** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the New Business Transaction Match Rule-Drupal window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. Check the check box in the Business Transaction Match Criteria section.
9. In the text field, enter the value against which to match the page callback name.
10. Select the condition to use to determine the match from the dropdown list. To reverse the condition, click the gear icon and check the NOT condition check box.
11. Click **Create Custom Match Rule**.

To modify a custom match rule for a Drupal entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 through 10 in [To configure a custom match rule for a Drupal entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

To remove a custom match rule for a Drupal entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.

3. From the Custom Match Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

#### Configure Exclude Rules for Drupal Entry Points

You can configure an exclude rule to exclude from detection any entry point that matches the configured match conditions. The match conditions are the same as those available for [configuring custom match rules](#).

Transactions excluded through exclude rules are not counted towards the business transaction limits.

#### To configure an exclude rule for a Drupal entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. Click the add icon to add a new exclude rule.
4. Select **Drupal** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the Exclude Business Transaction Match Rule-Drupal window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. Check the check box in the Business Transaction Match Criteria section.
9. In the text field, enter the value against which to match the page callback name.
10. Select the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.
11. Click **Create Exclude Rule**.

#### To modify an exclude rule for a Drupal entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to modify.

4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 through 10 in [To configure an exclude rule for a Drupal entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

**To remove an exclude rule for a Drupal entry point**

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

**Learn More**

- [Configure Transaction Detection for PHP](#)
- [Organizing Traffic as Business Transactions](#)
- [Match Rule Conditions](#)
- [Regular Expressions In Match Conditions](#)

#### **Configure PHP Wordpress Entry Points**

- [Default Detection and Naming for Wordpress Entry Points](#)
- [Configure Custom Match Rules for Wordpress Entry Points](#)
  - [To configure a custom match rule for a Wordpress entry point](#)
  - [To modify a custom match rule for a Wordpress entry point](#)
  - [To remove a custom match rule for a Wordpress entry point](#)
- [Configure Exclude Rules for Wordpress Entry Points](#)
  - [To configure an exclude rule for a Wordpress entry point](#)
  - [To modify an exclude rule for a Wordpress entry point](#)
  - [To remove an exclude rule for a Wordpress entry point](#)
- [Learn More](#)

By default, automatic transaction detection for Wordpress transactions is turned off. See [Configure Transaction Detection for PHP](#) for general information about turning on automatic detection in the PHP entry point window.

#### **Default Detection and Naming for Wordpress Entry Points**

By default, AppDynamics automatically names Wordpress transactions based on the Wordpress template name.

You may be seeing too many or too few Wordpress transactions. If you are not getting the visibility that you need with the default configuration, create one or more custom match rules for Wordpress entry points.

If you are seeing too many transactions, create exclude rules to prevent the agent from discovering the entry points that you do not need to monitor. Or create custom match rules that group several entry points into a single entry point.



If you are seeing too few transactions, review your custom match rules rules to make sure that they detect all the entry points that you want the agent to discover.

#### Configure Custom Match Rules for Wordpress Entry Points

You can configure a custom match rule that matches a portion of the page template name (such as "Begins with" or "Contains") rather than the default which matches the entire page template name ("Equals").

#### To configure a custom match rule for a Wordpress entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. Click the add icon to add a new custom match rule.
4. Select **Wordpress** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the New Business Transaction Match Rule-Wordpress window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. Check the check box in the Business Transaction Match Criteria section.
9. In the text field, enter the value against which to match the page template name.
10. Select the condition to use to determine the match from the dropdown list. To reverse the condition, click the gear icon and check the NOT condition check box.
11. Click **Create Custom Match Rule**.

#### To modify a custom match rule for a Wordpress entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 through 10 in [To configure a custom match rule for a Wordpress entry point](#). If you want to disable the rule, clear the Enabled check box.

6. Click **Save**.

To remove a custom match rule for a Wordpress entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

Configure Exclude Rules for Wordpress Entry Points

You can configure an exclude rule to exclude from detection any entry point that matches the configured match conditions. The match conditions are the same as those available for [configuring custom match rules](#).

Transactions excluded through exclude rules are not counted towards the business transaction limits.

To configure an exclude rule for a Wordpress entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. Click the add icon to add a new exclude rule.
4. Select **Wordpress** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the Exclude Business Transaction Match Rule-Wordpress window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. Check the check box in the Business Transaction Match Criteria section.
9. In the text field enter the value against which to match the page template name.
10. Select the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.
11. Click **Create Exclude Rule**.

New Exclude Business Transaction Match Rule - Wordpress

Name

Enabled ☒

Business Transaction Match Criteria

☒ Page Template Name **Contains**

To modify an exclude rule for a Wordpress entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 through 10 in [To configure an exclude rule for a Wordpress entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

To remove an exclude rule for a Wordpress entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

Learn More

- [Configure Transaction Detection for PHP](#)
- [Organizing Traffic as Business Transactions](#)
- [Match Rule Conditions](#)
- [Regular Expressions In Match Conditions](#)

## Configure PHP Command Line Interface (CLI) Entry Points






- [Detection for PHP CLI Entry Points](#)
- [PHP CLI Business Transactions](#)
- [Configure Custom Match Rules for PHP CLI Entry Points](#)
  - [To configure a custom match rule for a PHP CLI entry point](#)
  - [To modify a custom match rule for a PHP entry point](#)
  - [To remove a custom match rule for a PHP CLI entry point](#)
- [Configure Exclude Rules for PHP CLI Entry Points](#)
  - [To configure an exclude rule for a PHP CLI entry point](#)
  - [To modify an exclude rule for a PHP CLI entry point](#)
  - [To remove an exclude rule for a PHP CLI entry point](#)
- [Learn More](#)

Before configuring a PHP CLI entry point, make sure you have arranged to run the proxy daemon manually. This is required for monitoring PHP CLI. See [Special Procedures for PHP CLI](#) and [Running the PHP Proxy Daemon Manually](#).

## Detection for PHP CLI Entry Points

The App Agent for PHP automatically detects PHP CLI entry points if the PHP CLI entry point type

is enabled.

▼ Entry Points		
Type	Transaction Monitoring	Automatic Transaction Detection
 PHP Web	<input checked="" type="checkbox"/> Enabled	<a href="#">Configure Naming</a> <input type="checkbox"/> Use Virtual Host in Business Transaction names
 PHP CLI	<input checked="" type="checkbox"/> Enabled	
 PHP MVC	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Use Virtual Host in Business Transaction names
 Drupal	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Use Virtual Host in Business Transaction names
 Wordpress	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Use Virtual Host in Business Transaction names

#### PHP CLI Business Transactions

By default, AppDynamics automatically names PHP CLI transactions based on the last two segments of the script's directory path plus the name of the script.

The agent creates a business transaction instance every time the script is run.

If you are seeing too many transactions, create exclude rules to prevent the agent from discovering the transactions that you do not need to monitor. Or create custom match rules that group several entry points into a single entry point.

If you are seeing too few transactions, review your custom match rules rules to make sure that they detect all the PHP CLI entry points that you want the agent to discover.

#### Configure Custom Match Rules for PHP CLI Entry Points

You can configure a custom match rule that matches a portion of the script name (such as "Begins with" or "Contains" or "Reg Ex") rather than the default which matches the entire script name ("Equals"). The following rule matches a regular expression for scripts that start with "mysql." and end with "php".

#### To configure a custom match rule for a PHP CLI entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. Click the add icon to add a new custom match rule.
4. Select **PHP CLI** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the New Business Transaction Match Rule-PHP CLI window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. Check the check box in the Business Transaction Match Criteria section.
9. In the text field, enter the value against which to match the page callback name.
10. Select the condition to use to determine the match from the dropdown list. To reverse the condition, click the gear icon and check the NOT condition check box.
11. Click **Create Custom Match Rule**.

#### To modify a custom match rule for a PHP entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 through 10 in [To configure a custom match rule for a PHP CLI entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

#### To remove a custom match rule for a PHP CLI entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).

2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

#### Configure Exclude Rules for PHP CLI Entry Points

You can configure an exclude rule to exclude from detection any entry point that matches the configured match conditions. The match conditions are the same as those available for [configuring custom match rules](#).

Transactions excluded through exclude rules are not counted towards the business transaction limit of 50 transactions per agent or 200 transactions per application.

#### To configure an exclude rule for a PHP CLI entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. Click the add icon to add a new exclude rule.
4. Select **PHP CLI** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the Exclude Business Transaction Match Rule-PHP CLI window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. Check the check box in the Business Transaction Match Criteria section.
9. In the text field, enter the value against which to match the page callback name.
10. Select the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.
11. Click **Create Exclude Rule**.

For example, you may have a custom match rule that detects all the scripts in the /xyz directory but you want to exclude from detection the /xyz/a.php script.

#### To modify an exclude rule for a PHP CLI entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See steps 7 through 10 in [To configure an exclude rule for a PHP CLI entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

To remove an exclude rule for a PHP CLI entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. From the Exclude Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

Learn More

- [Configure Transaction Detection for PHP](#)
- [Running the PHP Proxy Daemon Manually](#)
- [Organizing Traffic as Business Transactions](#)
- [Match Rule Conditions](#)
- [Regular Expressions In Match Conditions](#)

#### Configure PHP Web Service Entry Points

- [Default Detection and Naming for PHP Web Service Entry Points](#)
- [Configure Custom Match Rules for PHP Web Service Entry Points](#)
  - [To configure a custom match rule for a PHP Web Service entry point](#)
  - [To modify a custom match rule for a PHP Web Service entry point](#)
  - [To remove a custom match rule for a PHP Web Service entry point](#)
- [Configure Exclude Rules for PHP Web Services Entry Points](#)
  - [To configure an exclude rule for a PHP Web Services entry point](#)
  - [To modify an exclude rule for a PHP Web Service entry point](#)
  - [To remove an exclude rule for a PHP Web Service entry point](#)

#### *New in 3.8.2*

By default, automatic transaction detection for PHP Web Services transactions is turned off. See [Configure Transaction Detection for PHP](#) for general information about turning on automatic detection in the PHP entry point window.

#### Default Detection and Naming for PHP Web Service Entry Points

By default, for Web Services frameworks the business transaction is named for the web service name and the operation name.

You may be seeing too many or too few PHP Web Service transactions. If you are not getting the visibility that you need with the default configuration, create one or more custom match rules for

PHP Web Service entry points.

If you are seeing too many transactions, create exclude rules to prevent the agent from discovering the entry points that you do not need to monitor. Or create custom match rules that group several entry points into a single entry point.

If you are seeing too few transactions, review your custom match rules rules to make sure that they detect all the entry points that you want the agent to discover.

#### Configure Custom Match Rules for PHP Web Service Entry Points

You can configure a custom match rule that matches a portion of the web service name and/or operation name (such as "Begins with" or "Contains") rather than the default which matches the entire name ("Equals").

The agent determines the match using only the configured criteria. So if you configure a rule based on matching a specific web service name but do not configure the operation name, the agent does not consider the operation name in evaluating the match.

To configure a custom match rule for a PHP Web Service entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. Click the add icon to add a new custom match rule.
4. Select **PHP Web Service** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the New Business Transaction Match Rule - PHP Web Services window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.
8. In the Business Transaction Match Criteria section, for each match condition that you want to specify, do the following:
  - a. Check the check box for the segment for which you are configuring a match.
  - b. In the text field, enter the value against which to match.



- c. Select the condition to use to determine the match from the dropdown list.  
To reverse the condition, click the gear icon and check the NOT condition check box.

9. Click **Create Custom Match Rule**.

To modify a custom match rule for a PHP Web Service entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to modify.
4. Double-click the rule or click the edit icon.
5. Modify the rule as needed. See step 8 in [To configure a custom match rule for a PHP Web Service entry point](#). If you want to disable the rule, clear the Enabled check box.
6. Click **Save**.

To remove a custom match rule for a PHP Web Service entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Custom Match Rules section.
3. From the Custom Match Rules list, select the rule that you want to remove.
4. Click the delete icon.
5. In the Confirm Delete window click **OK**.

#### Configure Exclude Rules for PHP Web Services Entry Points

You can configure an exclude rule to exclude from detection any entry point that matches the configured match conditions. The match conditions are the same as those available for [configuring a custom match rule](#).

Transactions excluded through exclude rules are not counted towards the business transaction limit.

To configure an exclude rule for a PHP Web Services entry point

1. Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
2. Scroll down to the Exclude Rules section.
3. Click the add icon to add a new exclude rule.
4. Select **PHP Web Services** from the Entry Point Type dropdown list.
5. Click **Next**.
6. In the New Exclude Business Transaction Match Rule - PHP Web Service window, enter a name for the rule.
7. Check the Enabled check box to enable the rule.

8. In the Business Transaction Match Criteria section, for each match condition that you want to specify, do the following:

- Check the check box for the segment for which you are configuring a match.
  - In the text field enter the value against which to match.
  - Select the condition to use to determine the match from the dropdown list.
- To reverse the condition, click the gear icon and check the NOT condition check box.

9. Click **Create Exclude Rule**.

**To modify an exclude rule for a PHP Web Service entry point**

- Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
- Scroll down to the Exclude Rules section.
- From the Exclude Rules list, select the rule that you want to modify.
- Double-click the rule or click the edit icon.
- Modify the rule as needed. See step 8 in [To configure an exclude rule for a PHP Web Service entry point](#). If you want to disable the rule, clear the Enabled check box.
- Click **Save**.

**To remove an exclude rule for a PHP Web Service entry point**

- Access the PHP transaction detection instrumentation window. See [Accessing Transaction Detection](#).
- Scroll down to the Exclude Rules section.
- From the Exclude Rules list, select the rule that you want to remove.
- Click the delete icon.
- In the Confirm Delete window click **OK**.

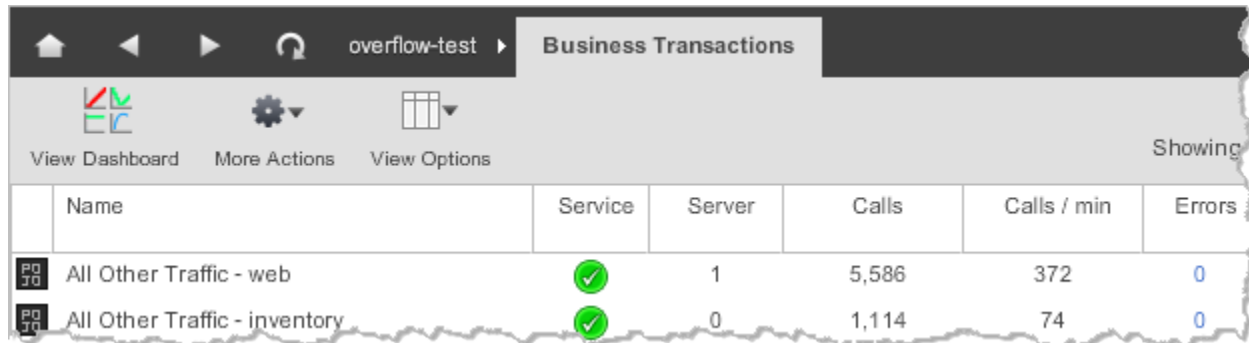
## Calibrate PHP Entry Point Detection

- [Process](#)
- [1 Confirm Business Relevance](#)
- [2 Exclude Business Transactions](#)

- [3 Combine Business Transactions](#)
- [4 Split PHP Web Business Transactions](#)
- [5 Delete Unnecessary Business Transactions](#)
- [Learn more](#)

Sometimes you need to fine-tune your entry point detection configuration, for example:

- When you are not seeing expected business transactions
- When you see All Other Traffic business transactions in your business transaction list, as shown here:



The screenshot shows the AppDynamics interface with the 'Business Transactions' tab selected. The table lists two entries under 'All Other Traffic': 'All Other Traffic - web' and 'All Other Traffic - inventory'. Both entries have a green checkmark in the 'Service' column, indicating they are correctly identified. The 'Server' column shows '1' for web and '0' for inventory. The 'Calls' column shows 5,586 for web and 1,114 for inventory. The 'Calls / min' column shows 372 for web and 74 for inventory. The 'Errors' column shows 0 for both.

Name	Service	Server	Calls	Calls / min	Errors
All Other Traffic - web	✓	1	5,586	372	0
All Other Traffic - inventory	✓	0	1,114	74	0

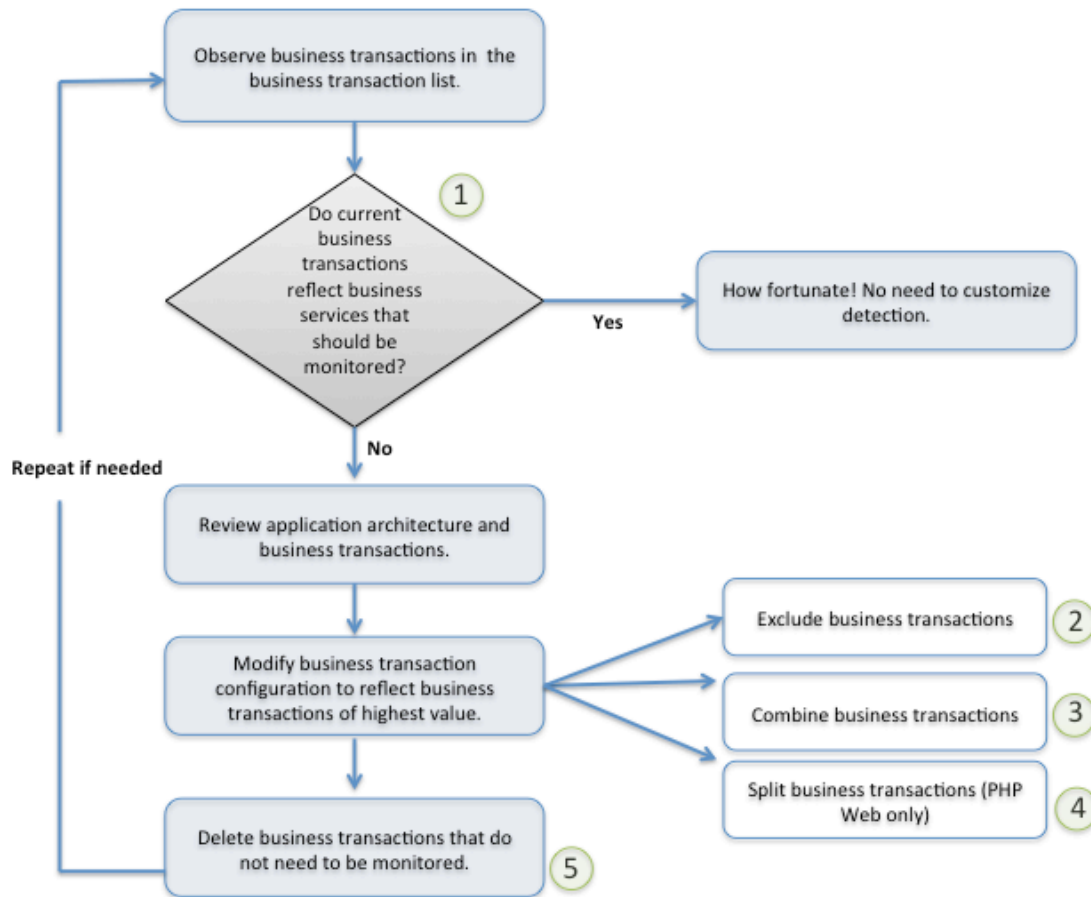
To analyze your current configuration try one or more of the following tactics:

- Confirm that you have the right entry point for the start of your business transaction, see [Organizing Traffic as Business Transactions](#).
- Review the application architecture and make sure the expected business transactions are not part of another transaction in an upstream tier.
- Review the existing rules and if necessary, modify them using one or more of the following techniques:
  - Combine business transactions using custom match rules.
  - Exclude some entry points.
  - Create and fine-tune custom match rules to define entry points precisely.

There is a default limit of 50 business transactions per agent and 200 business transactions per business application. If you need to increase the limits, set it in the max-business-transactions property in <php-agent-root>/proxy/conf/app-agent-config.xml. There is a hard limit of 100 business transactions per agent.

## Process

The following flow chart suggests a process for analyzing your entry point detection scheme.



Confirm

### 1 Confirm Business Relevance

The first step in determining the entry point configuration that is right for your application is to decide which transactions you want to monitor. Because business transactions begin with specific entry points, this may require talking to application developers and architects to confirm the correct set of entry points. Be sure you are measuring the right things.

Once you know what you want to monitor, you can examine the business transactions currently being detected and determine your next steps. This topic describes several techniques for modifying your entry point configuration.

Exclude

## 2 Exclude Business Transactions

Use exclude rules to prevent the agent from detecting entry points for business transactions that you do not need to monitor.

Here are some scenarios where exclude rules might be useful.

### ***Business Transaction Discovery at the Next Layer of Application Component Logic***

By default, the agent detects PHP MVC requests based on the module\*:**c ontroller\*:action**. When an incoming PHP MVC request starts at some control logic in your code that triggers different business logic based on the action, you may prefer to use just the action as the business transaction name. You can create an PHP MVC exclude rule using the match criteria on the action name to prevent certain business transactions from being discovered.

The following rule excludes entry points in which the action starts with **indexAction**.

### ***Use an Exclude Rule as a Filter***

You can use an exclude rule as a filter to allow the eligible requests and ignore everything else.

For example, you want to use default discovery rules to identify the correct entry points. Your application receives URI ranges that start with **/a, /b ... /z**, but you want to monitor only URIs only that start with **/a** and **/b**. Create an exclude rule that matches on Doesn't Start With /a or /b as shown here:

[Combine](#)

### 3 Combine Business Transactions

When you need to reduce the number of entry points that are automatically discovered, you can use a custom match rule to combine multiple potential entry points into a single entry point.

For example, if you have PHP MVC entry points that are automatically detected as

- catalog1:productA:view
- catalog2:productB:view
- catalog2:productA:view
- catalog3:productD:view

and you want just a single transaction for catalog, create a custom match rule matching only on "catalog".

New Business Transaction Match Rule - PHP MVC

Name: Catalog Rule

Enabled: ☒

Business Transaction Match Criteria

Criteria	Operator	Value
<input checked="" type="checkbox"/> Controller	Contains	catalog
<input type="checkbox"/> Action	Equals	
<input type="checkbox"/> Module	Equals	

Buttons: Cancel, Create Custom Match Rule

Or if you have PHP Web entry points that are detected as

- /clothing/girls
- /clothing/boys
- /clothing/women
- /clothing/men

and you want just a single transaction for all clothing, create a custom match rule matching only on "clothing".

New Business Transaction Match Rule - PHP Web

Name: Clothing Rule

Enabled: ☒

Priority: 0

Transaction Match Criteria

Split Transactions Using Request Data

Criteria	Operator	Value
<input type="checkbox"/> Method		
<input checked="" type="checkbox"/> URI	Contains	clothing

Or you want the agent to ignore case in the URI, so that **/appdynamics.com** and **/APPDYNAMICS.com** are detected as a single transaction, not two transactions.

New Business Transaction Match Rule - PHP Web

Name: Ignore Case Rule

Enabled: ☒

Priority: 0

Transaction Match Criteria

Split Transactions Using Request Data

Method: ☐

URI: ☒ Matches Reg Ex: (?)url

HTTP Parameter: ☐ Check for parameter value

After you have created a custom match rule, delete the old business transactions that were originally discovered. Deleting a business transaction deletes all historical data regarding that transaction. When the agent detects the transaction again, it uses the new custom match rule rather than the default discovery rule. See [5 Delete Unnecessary Business Transactions](#).

Split

#### 4 Split PHP Web Business Transactions

Using a dynamic value to customize business transaction discovery is called transaction splitting. Transaction splitting allows you to fine-tune detection or exclusion based on a parameter or user data. For the PHP agent, this feature is available for PHP Web type entry points only.

For example, the agent detects the URI **/company/careers** but you want to track separate business transactions for **/careers/engineering**, **/careers/marketing**, **/careers/sales**, and so on. Create a custom match rule with transaction splitting.

**New Business Transaction Match Rule - PHP Web**

Name: Careers Match Rule

Enabled: ☒

Priority: 0

**Transaction Match Criteria** | **Split Transactions Using Request Data**

☒ Split Transactions using request data

☐ Use the first  segments in Transaction names

☐ Use the last  segments in Transaction names

☒ Use URI segment(s) in Transaction names

Segment Numbers: 2,3 Enter a comma separated list of parameter numbers (e.g. 1,3,4)

☐ Use a parameter value in Transaction names

Parameter Name:

Or you want to split a URI that contains a parameter as in **/myonline/store/shop?productid=dvd**, **/myonline/store/shop?productid=cd**, **/myonline/store./shop?productid=book**.

**New Business Transaction Match Rule - PHP Web**

☒ Split Transactions using request data

☐ Use the first  segments in Transaction names

☐ Use the last  segments in Transaction names

☐ Use URI segment(s) in Transaction names

Segment Numbers: 2,3 Enter a comma separated list of parameter numbers (e.g. 1,3,4)

☒ Use a parameter value in Transaction names

Parameter Name: productid

☐ Use a header value in Transaction names



A rectangular button with rounded corners, a light blue gradient, and a thin grey border. The word "Delete" is written in a bold, green, sans-serif font in the center.

### 5 Delete Unnecessary Business Transactions

After you have modified the business transaction discovery configuration, you should delete the old transactions that were discovered using the old discovery rules.

If you delete a business transaction and you have not changed the entry point configuration, the agent will rediscover it. However, if you have modified the transaction detection rules then delete the old business transaction, it will not be rediscovered.

You delete business transactions from the business transaction list in the AppDynamics console. See the information about the **Delete** operation at [Business Transactions List Operations](#).

#### Learn more

- [Configure Business Transaction Detection](#)
- [All Other Traffic Business Transaction](#)
- [Match Rule Conditions](#)
- [Regular Expressions In Match Conditions](#)
- [PHP Entry Points](#)

### Configure Error Detection for PHP

- [Accessing Error Detection Configuration for the PHP Agent](#)
  - [To Access Error Detection Configuration](#)
- [Error Detection Threshold for the PHP Agent](#)
- [Reporting Errors in Business Transactions for the PHP Agent](#)
  - [To configure basic error detection for the PHP agent](#)
- [Configuring the PHP Agent to Ignore Exceptions and Log Messages](#)
  - [Configuring Exceptions to Ignore](#)
    - [To configure the PHP agent to ignore an exception](#)
  - [Configuring Logged Messages to Ignore](#)
    - [To configure the PHP agent to ignore a log message](#)

The App Agent for PHP instruments the PHP reporting facility. PHP applications can use `trigger_error` to report errors through that facility. PHP extensions and PHP itself can also use the PHP facility for reporting errors.

However, the App Agent for PHP does not report errors that occur during an exit call but instead creates a synthetic exception for such errors.

You can configure the threshold for log messages and exceptions that the PHP agent reports as errors. You can also define exceptions that should not be included in the business transaction

error count and log messages that should not be reported to the controller.

These configurations help you reduce the number of errors that the agent reports so you see just those that are most useful for monitoring and troubleshooting.

## Accessing Error Detection Configuration for the PHP Agent

### To Access Error Detection Configuration

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

## Error Detection Threshold for the PHP Agent

To control the number of detected errors, you can choose the minimum threshold of errors that the agent should report from Error, Warning, and Notice.

If you select Error, the agent reports only messages and exceptions marked Error.

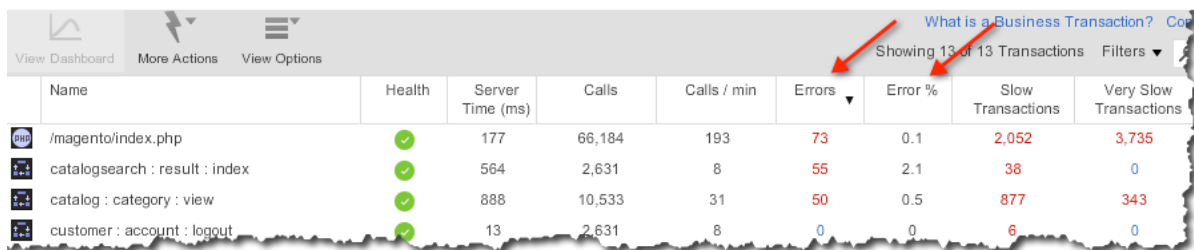
If you select Warning, the agent reports only messages and exceptions marked Error and Warning.

If you select Notice, the agent reports messages and exceptions marked Error, Warning and Notice.

## Reporting Errors in Business Transactions for the PHP Agent

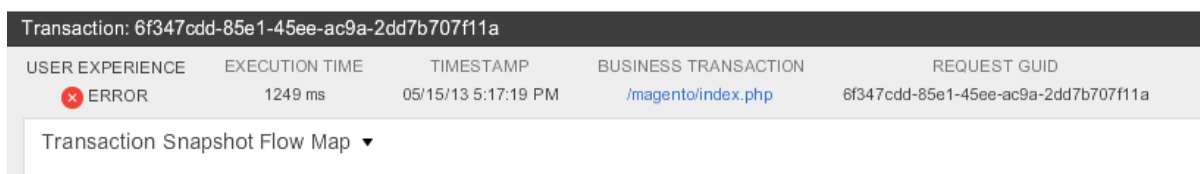
Errors in business transactions are displayed in a variety of places in the AppDynamics console:

- In the transaction scorecards in various dashboards
- In the Errors trend graph on the business transaction dashboard
- In the business transaction lists



Name	Health	Server Time (ms)	Calls	Calls / min	Errors	Error %	Slow Transactions	Very Slow Transactions
/magento/index.php	✓	177	66,184	193	73	0.1	2,052	3,735
catalogsearch : result : index	✓	564	2,631	8	55	2.1	38	0
catalog : category : view	✓	888	10,533	31	50	0.5	877	343
customer : account : logout	✓	13	2,631	8	0	0	6	0

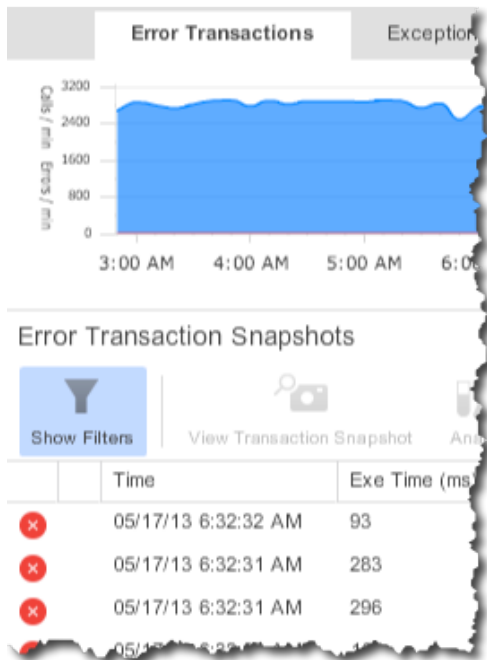
- In error transaction snapshots



Transaction: 6f347cdd-85e1-45ee-ac9a-2dd7b707f11a				
USER EXPERIENCE	EXECUTION TIME	TIMESTAMP	BUSINESS TRANSACTION	REQUEST GUID
✗ ERROR	1249 ms	05/15/13 5:17:19 PM	/magento/index.php	6f347cdd-85e1-45ee-ac9a-2dd7b707f11a

Transaction Snapshot Flow Map ▾

- In the Error Transactions tab accessed from the **Troubleshoot->Errors** item



You can configure whether you want log messages and exceptions within the error threshold to be marked as transaction errors and counted by checking or clearing the Mark Business Transaction as error check box.

✖ Error Detection Using Logged Exceptions or Messages

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

☒ Detect Errors Using threshold of: **Error** ▼

☒ Mark Business Transaction as error

When error detection is enabled, if the Mark Business Transaction as error check box is clear, the agent reports exceptions and log messages to the controller for all errors within the configured threshold, but it does not increment the error counts and the errors per minute counts reported for the affected business transactions.

If the Mark Business Transaction as error check box is checked, detected log messages and exceptions cause the error counts to be incremented. Transaction snapshots generated for these errors are error transaction snapshots.

You can further refine which errors cause business transaction error counts to be incremented through ignore rules. See [Configuring Exceptions to Ignore](#).

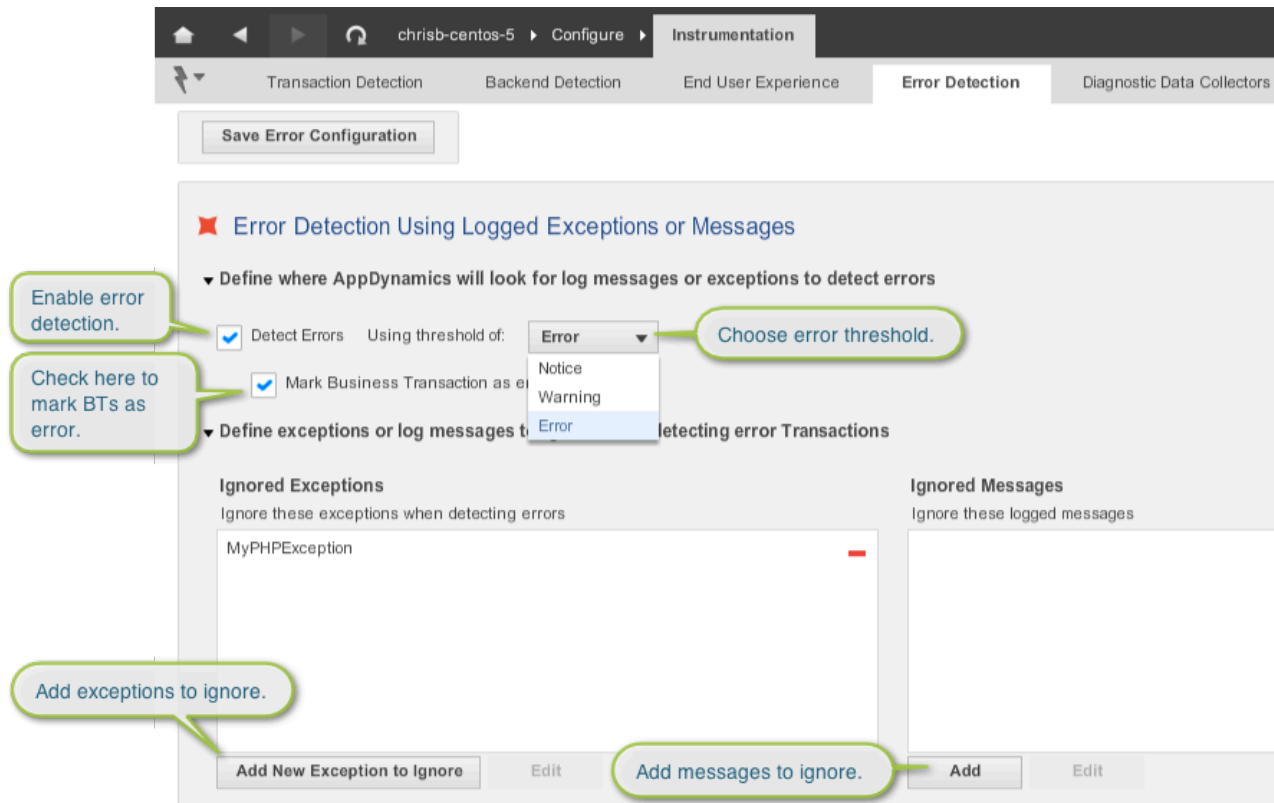
To configure basic error detection for the PHP agent

1. Check the Detect Error check box to enable error detection. Clear the check box to disable error detection.
2. If error detection is enabled, select the minimum threshold from the dropdown list.

Errors below the threshold are not reported.

3. If you want the agent to count the business transaction that caused the error as an error transaction, check the Mark Business Transaction as error check box. If you do not want the business transactions with errors to be counted as error transactions, clear this check box.

4. Click **Save Error Configuration** in the top left corner of the window.



## Configuring the PHP Agent to Ignore Exceptions and Log Messages

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

### Configuring Exceptions to Ignore

You can direct the agent to ignore an exception only when the exception message contains a certain string or does not contain the string.

The ignored exception is not really "ignored". This configuration just prevents the agent from incrementing business transaction error metrics caused by exceptions that have been configured to be ignored. When an ignored exception occurs, AppDynamics does not count the business transaction in which the exception occurred as an error transaction. But it still detects the exception, logs the exception, and increments the exception count. The exception is still displayed in the **Exceptions** subtab of the Errors tab of the tier and node dashboards and in the Summary and Error Details sections of any transaction snapshot that was in progress when the exception occurred.

Ignoring an exception simply means that AppDynamics does not increment the business transaction error count for those exceptions.

To configure the PHP agent to ignore an exception

1. Click **Add New Exceptions to Ignore** under the Ignored Exceptions list.
2. Enter the fully-qualified class names of exceptions to ignore, separated by colons.

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

**Exception, or Exception Chain**

Enter fully qualified Class Names for Exceptions separated by :

PHPExcelException

**Match Condition for Exception Message**

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

Exception Message **Contains** Access

Cancel Save

Exception chains are supported. If you specify "A:B" as the exception chain, the agent matches any chain where B is in the chain of A, even if B is not an immediate cause of the exception.

3. Optionally, further qualify the exceptions to ignore by configuring a match condition for the exception message. Enter the string to match and the qualifier from the dropdown list (Equals, Contains, Starts with, Ends with, Matches Reg Ex).

The RegEx operation uses PHP's built-in PCRE regular expression engine and requires the same syntax, including delimiters (for example: /^Foo/). See the [PCRE Manual](#) for help matching a regular expression.

Click the gear icon to configure the NOT condition for the match.

☐ NOT Condition

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

4. Click **Save**.
5. Click **Save Error Configuration** in the top left corner of the window.

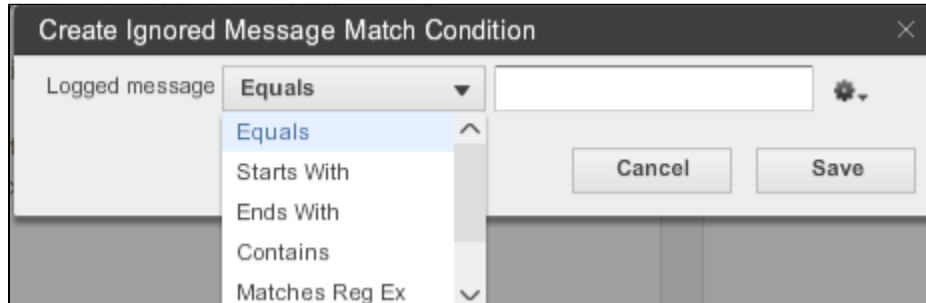
You can later edit or remove the ignore rule by clicking it in the Ignored Exceptions list.

### Configuring Logged Messages to Ignore

When a log message that has been configured to be ignored is logged, the PHP agent does not report the message to the controller.

To configure the PHP agent to ignore a log message

1. Click **Add** under the Ignored Messages list.
2. Enter the string to be matched in the message and the qualifier from the dropdown list (Equals, Contains, Starts with, Ends with, Matches Reg Ex).



The RegEx operation uses PHP's built-in PCRE regular expression engine and requires the same syntax, including delimiters (for example: `/^Foo/`). See the [PCRE Manual](#) for help matching a regular expression.

Click the gear icon to configure the NOT condition for the match.

3. Click **Save**.
4. Click **Save Error Configuration** in the top left corner of the window

## Configure Call Graphs for PHP

- [Call Graph Settings](#)
  - [To access call graph configuration screens](#)
  - [Call Graph Granularity](#)
  - [SQL Capture Settings](#)
    - [To configure SQL capture](#)
- [Learn More](#)

This topic describes how to configure call graphs in the PHP agent.

## Call Graph Settings

The Call Graph Settings window lets you configure thresholds that affect performance and how much detail about SQL statements to capture.

To access call graph configuration screens

1. In the left navigation pane, click **Configure -> Instrumentation**.
2. Click the Call Graph Settings tab.
3. Click the PHP Call Graph Settings subtab.

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

## Call Graph Granularity

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

- **For methods:** To ensure low performance overhead, choose a threshold in milliseconds for method execution time. Methods taking *less* than the time specified here are filtered out of the call graphs.
- **For SQL calls:** You can also specify a threshold for SQL queries. SQL queries taking *more* than the specified time in milliseconds are captured in the call-graphs.

To exclude internal functions from the call graphs, check the Exclude internal functions check box.

### SQL Capture Settings

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

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

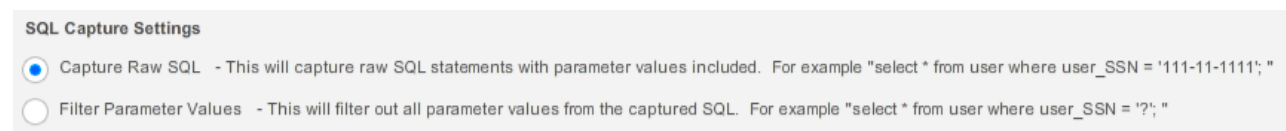
Replacing the literals in a query with parameter markers in this way is called normalizing the query.

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

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

#### To configure SQL capture

1. In the **Call Graph Settings** tab, scroll down to the SQL Capture Settings section.
2. Select one of the following:
  - **Capture Raw SQL:** Select this option to see raw SQL data (this captures raw SQL data along with the parameter values). Raw SQL data includes prepared statement bind variables or raw statements. By default, the private SQL data and queries that take less than 10 ms are not captured.
  - **Filter Parameter values:** Select this option to filter all literal values (constants) out of the captured query.



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

### Learn More

- [Call Graphs](#)
- [Transaction Snapshots](#)

## Monitor PHP Applications

### Distributed Transactions for PHP

Distributed transaction tracing allows you to trace the performance of a business transaction

across multiple tiers. It provides metrics for the time spent in all the tiers through which the business transaction passes and the ability to get call graphs displaying activity for all the tiers. See [Organizing Traffic as Business Transactions](#) and [Measure Distributed Transaction Performance](#) for general information.

The tiers in a distributed transaction can be heterogeneous. All combinations of PHP, Java, .NET, and Node.js tiers are supported, whether they are an originating, continuing, or terminating tier in the transaction.

## Monitor PHP Backends

A backend is an entity in the AppDynamics model that the app agent does not instrument directly, but traffic flows to it. The App Agent for PHP monitors flows to HTTP exit points, database exit points and cache exit points. See [Supported Environments and Versions for PHP](#) for the current list of supported backends.

You cannot configure detection and naming for PHP backends.

For general information about monitoring backends see:

- [Backend Monitoring](#)
- [Monitor Databases](#)
- [Monitor Remote Services](#)

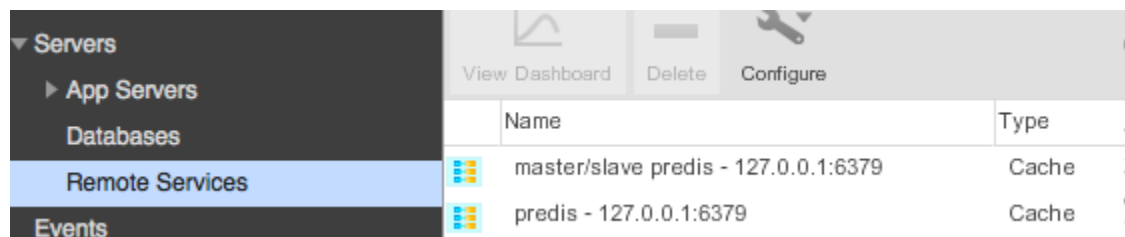
For special information about PHP backends see:

## Monitor Predis Backends

- [Predis Dashboard](#)
- [Learn More](#)

The App Agent for PHP monitors exit calls from an app server to a Predis backend.

To see the list of Predis backends, click **Servers->Remote Services**.



View Dashboard Delete Configure		
	Name	Type
	master/slave predis - 127.0.0.1:6379	Cache
	predis - 127.0.0.1:6379	Cache

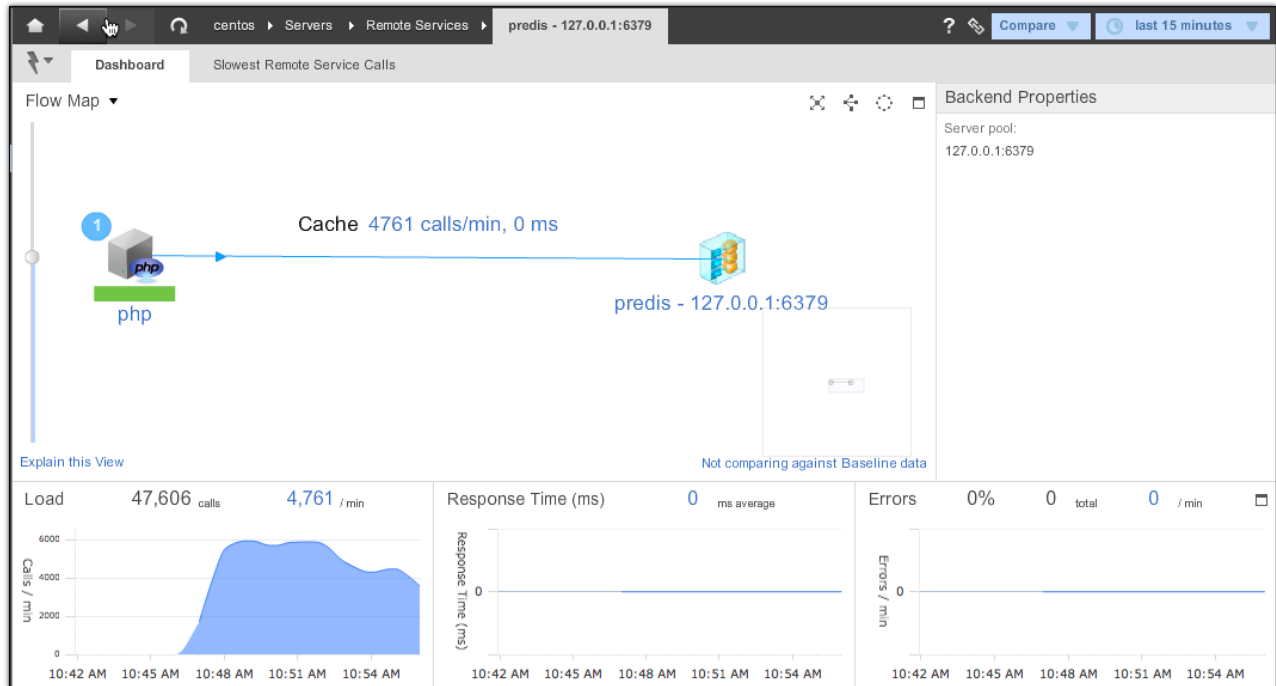
### Predis Dashboard

The dashboard for a Predis backend displays:

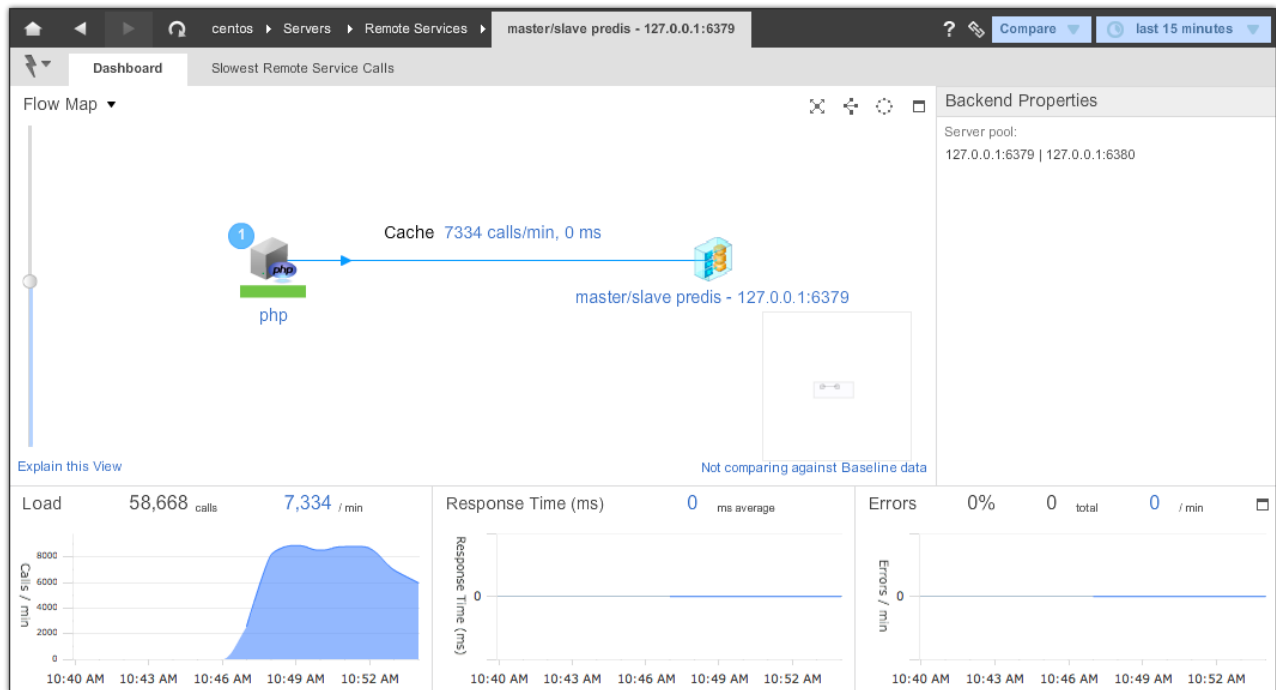
- a flow map showing the flow from the app server to the backend
- the backend properties: host and port
- graphs of the key performance indicators (KPIs) for flows to the backend

To see a backend dashboard, select it in the remote services list and either double-click or click **View Dashboard**.





The app agent allows you to monitor flows to master-slave replication configurations as well as single-server configurations. The dashboard for a master-slave Predis configuration displays a single node for the entire configuration. The backend properties include the *host:port* of the master separated by a pipe character (|) from a comma-separated list of *host:port* entries for the slaves.



The master/slave backend is tracked as a single server pool just as in a single-server configurations configuration. The agent does not provide separate call graphs for flows to multiple slaves.

## Learn More

- [Monitor Remote Services](#)

## Monitor RabbitMQ Backends for PHP

- [RabbitMQ Dashboard](#)
- [RabbitMQ Monitoring Extension](#)
- [Learn More](#)

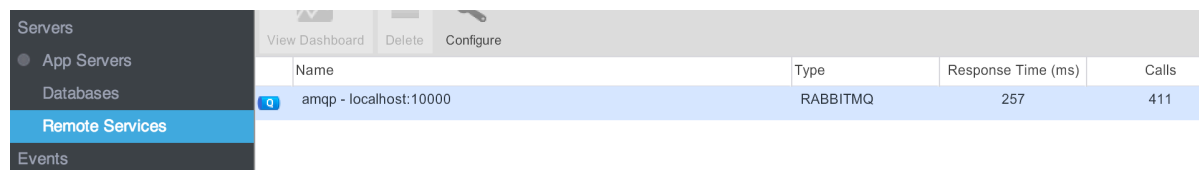
The App Agent for PHP monitors messages sent from a PHP tier into a RabbitMQ backend.

The App Agent for PHP does not monitor messages received by a PHP tier from a RabbitMQ backend.

However, in a distributed transaction, the App Agents for Java and .NET can monitor messages received from a RabbitMQ backend when the messages originated from a PHP tier.

The App Agent for PHP RabbitMQ support requires the [amqp extension](#).

To see the list of RabbitMQ backends, in the left navigation pane click **Servers->Remote Services**.



Name	Type	Response Time (ms)	Calls
amqp - localhost:10000	RABBITMQ	257	411

## RabbitMQ Dashboard

The dashboard for a RabbitMQ backend displays:

- a flow map showing the flow between app servers and the backend
- the backend properties: exchange, host, port, and routing key
- graphs of the key performance indicators (load, average response time, errors) for flows to the backend. To see a backend dashboard, select it in the remote services list and either double-click or click **View Dashboard**.



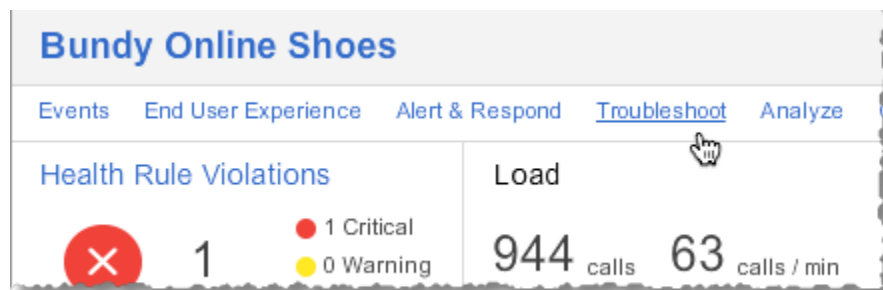
### Troubleshooting Your Application

For common troubleshooting scenarios see:

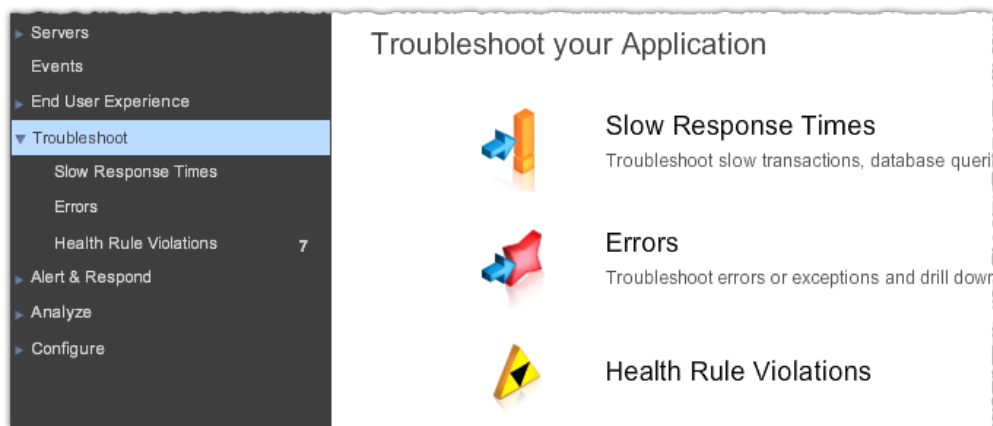
- [Troubleshoot Slow Response Times for PHP](#)
- [Troubleshoot Errors for PHP](#)
- [Troubleshoot Health Rule Violations](#)

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

- From the All Applications dashboard click the Troubleshoot link.



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



For more information about troubleshooting see:

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

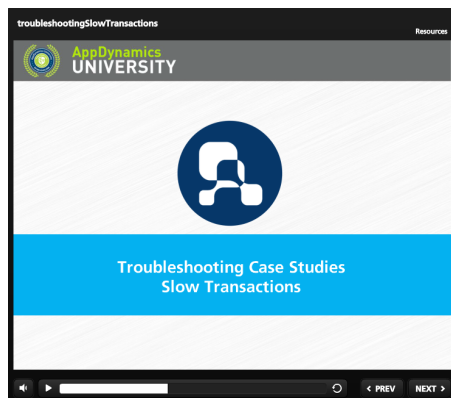
- [Transaction Snapshots](#)
- [Call Graphs](#)
- [Diagnostic Sessions](#)
- [Analyze](#)

## Troubleshoot Slow Response Times for PHP

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

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

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



## Slow and Stalled Transactions

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

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

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

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

### To troubleshoot slow and stalled transactions

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

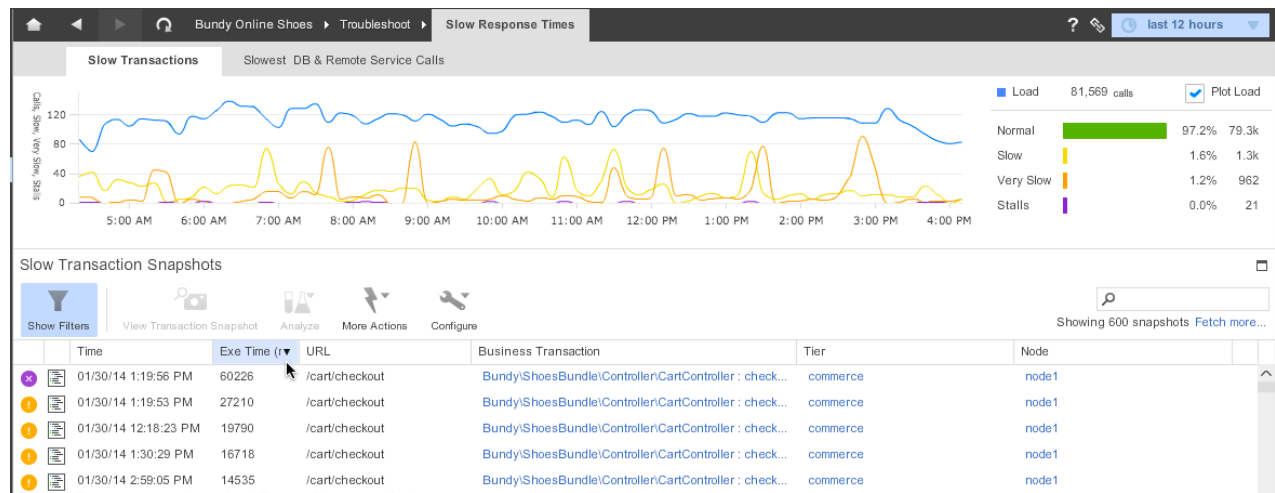
You can also access this information from tabs in the various dashboards.

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

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

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

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



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

## Slow Database and Remote Service Calls

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

### To troubleshoot slow database and remote service calls

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

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

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

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

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

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

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

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

See [Transaction Snapshots](#).

## Learn More

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

## Troubleshoot Errors for PHP

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

Identifying and troubleshooting errors in your application.

## Error Transactions and Exceptions

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

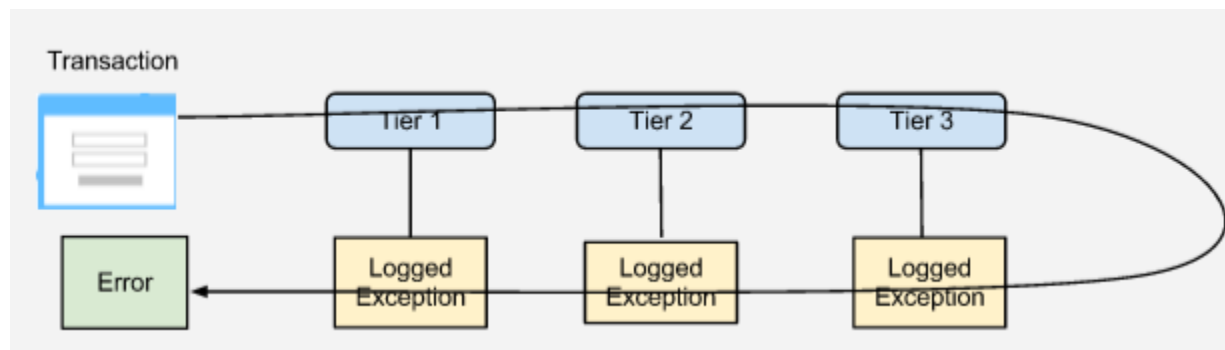
- A runtime error reported by the PHP server and captured by the agent. These include Fatal

Errors, Warning and Notices thrown by the PHP runtime. These types of errors do not provide a stack trace.

- Certain exceptions thrown by the application. These include unhandled exceptions (when an exception is thrown and there is no **try** block) and handled exceptions during an exit call (when there is a **try** block during an exit call). These types of errors do provide a stack trace.

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

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



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

#### To Troubleshoot Error Transactions

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

The error viewer opens.

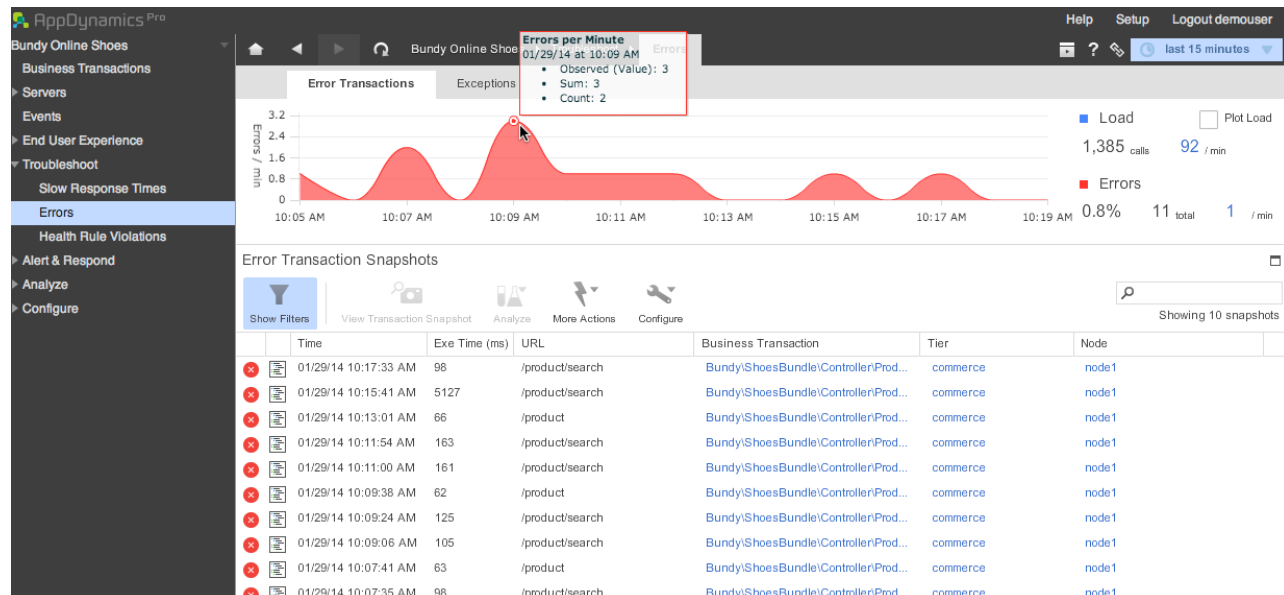
2. Click the **Error Transactions** tab if it is not already selected.
3. From the time range drop-down menu select the time range for which you want to view information about error transactions.

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

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

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



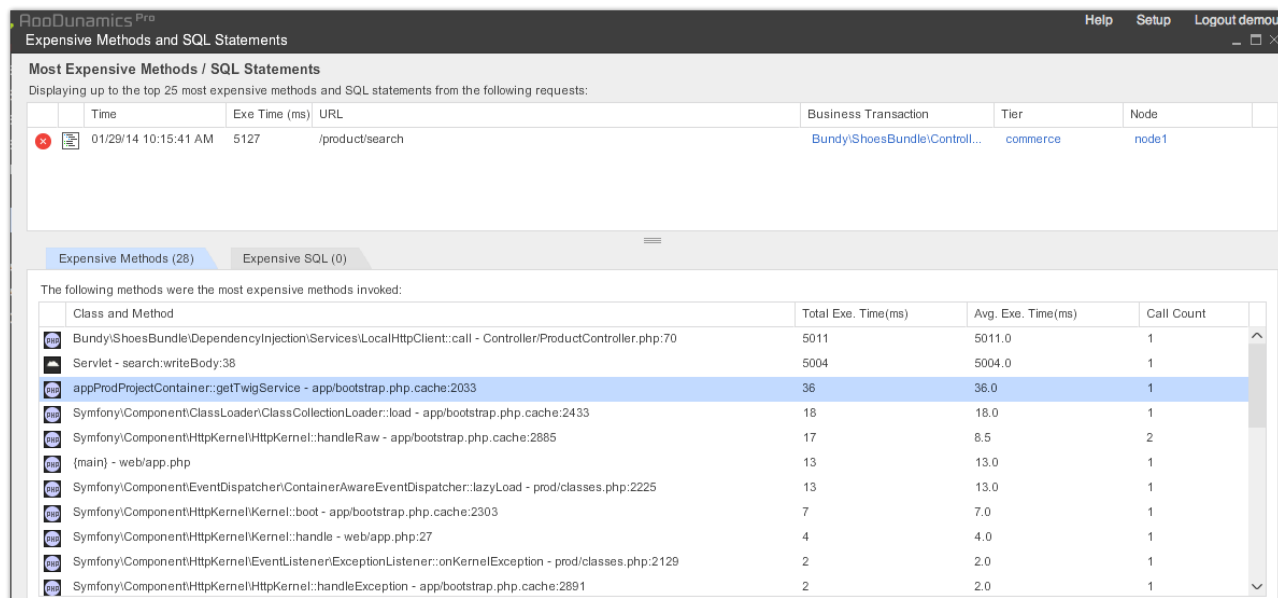


4. The error transaction snapshots are listed in the lower part of the viewer. To filter this list click **Show Filters** and select the filter criteria.

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

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

The Most Expensive Methods / SQL Statements viewer opens.



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

To Troubleshoot Exceptions

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

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

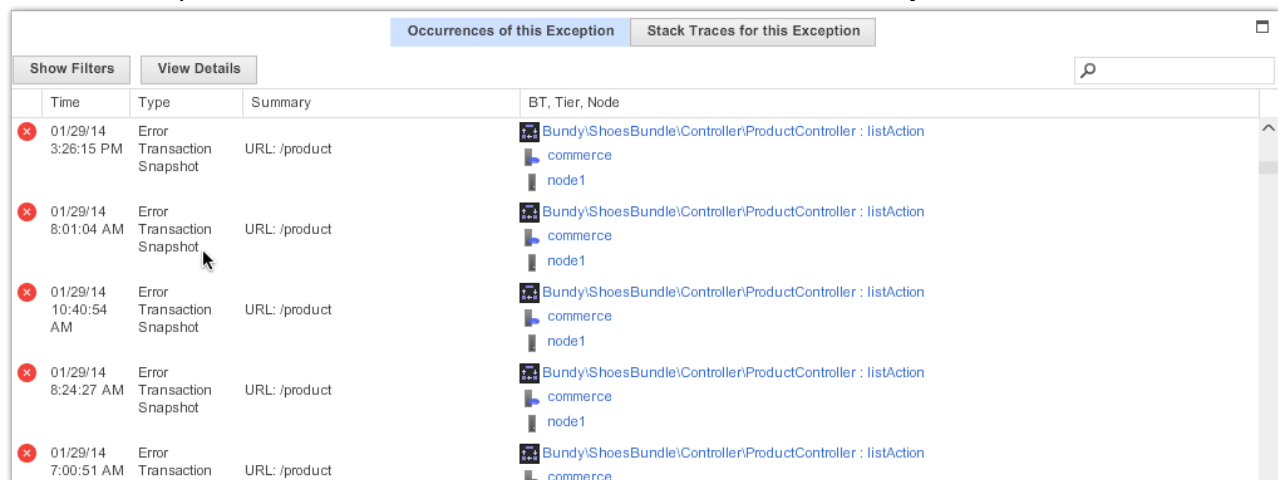
The exceptions list is displayed in the lower panel.

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

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

4. To view transaction snapshots for an exception:

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



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

- b. Select a snapshot from the list.

- c. Click **View Details**.

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

5. To view a stack trace for an exception:

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

- b. Click an exception in the left panel.

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

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

## Learn More

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

## Tutorials for PHP

See also:

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

## First Time Using the App Agent for PHP

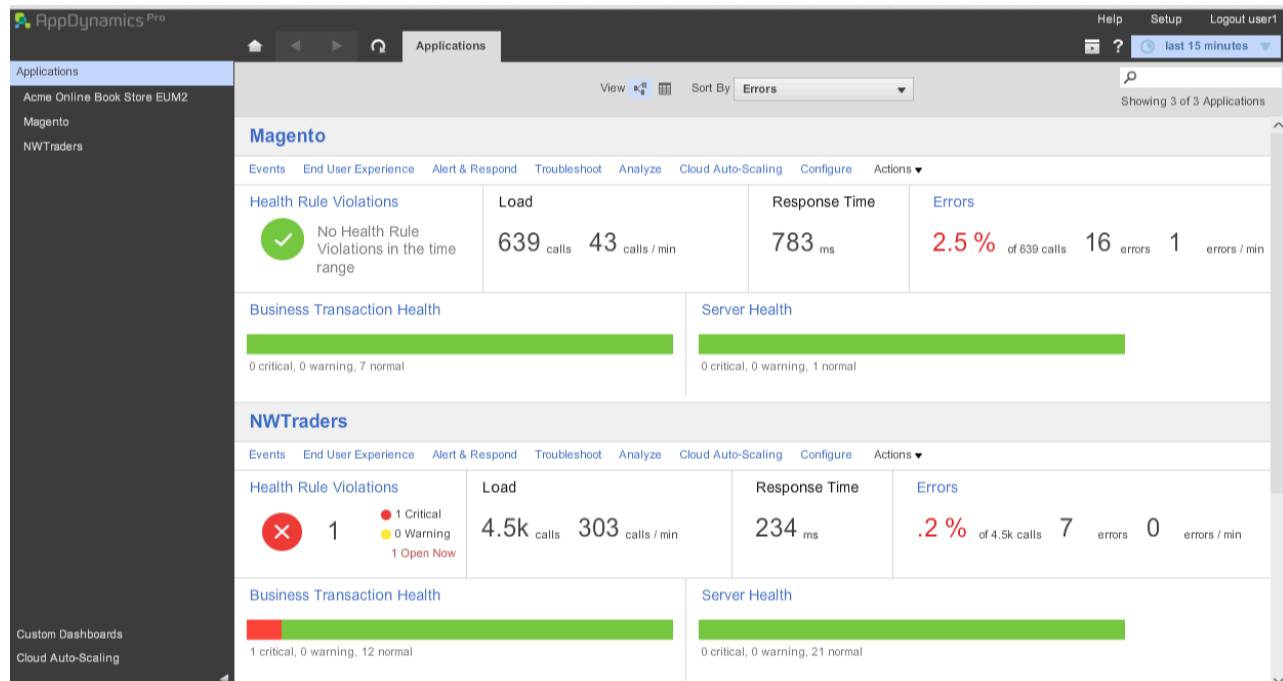
- [All Applications Dashboard](#)
- [Application Dashboard](#)
  - [Time Range](#)
  - [Flow Map and KPIs](#)
  - [Events](#)
  - [Transaction Scorecard](#)
  - [Exceptions and Errors](#)

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:

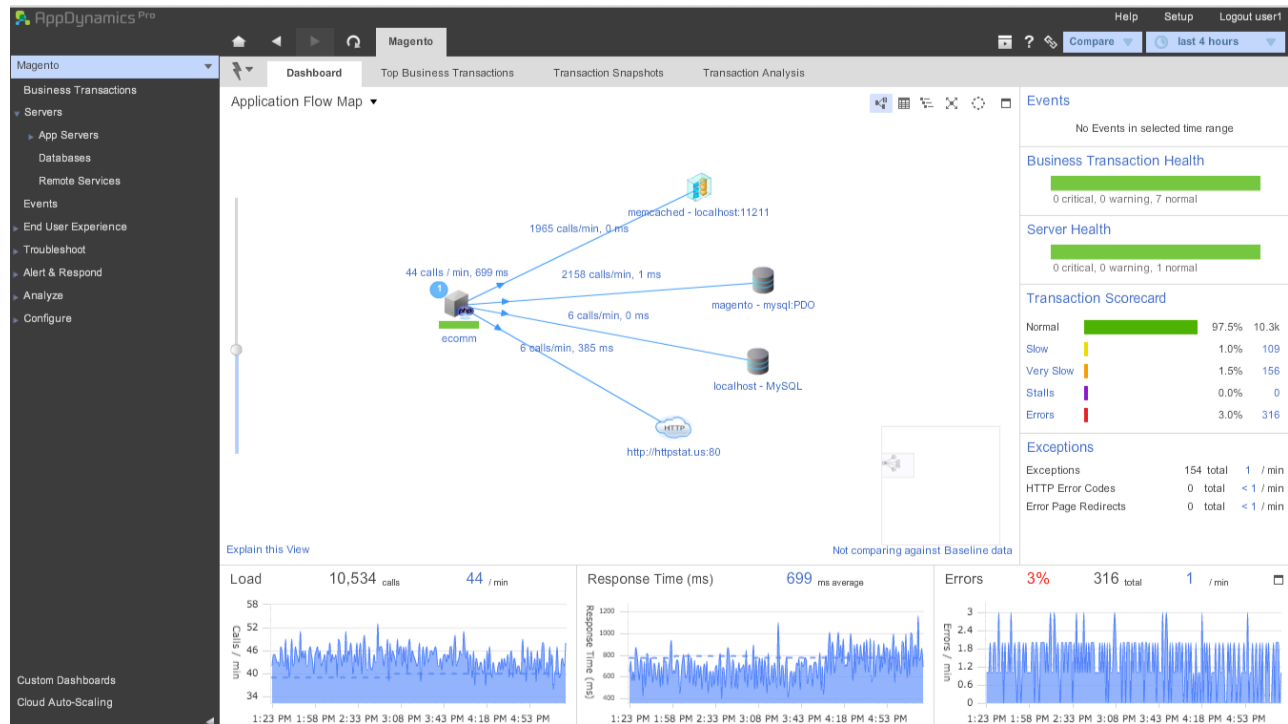
**Health Rule Violations:** AppDynamics lets you [define a health rule](#), which consists of a condition or a series of conditions based on metrics exceeding predefined thresholds or dynamic baselines. You can then use health rules in [policies](#) to automate optional remedial actions to take if the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

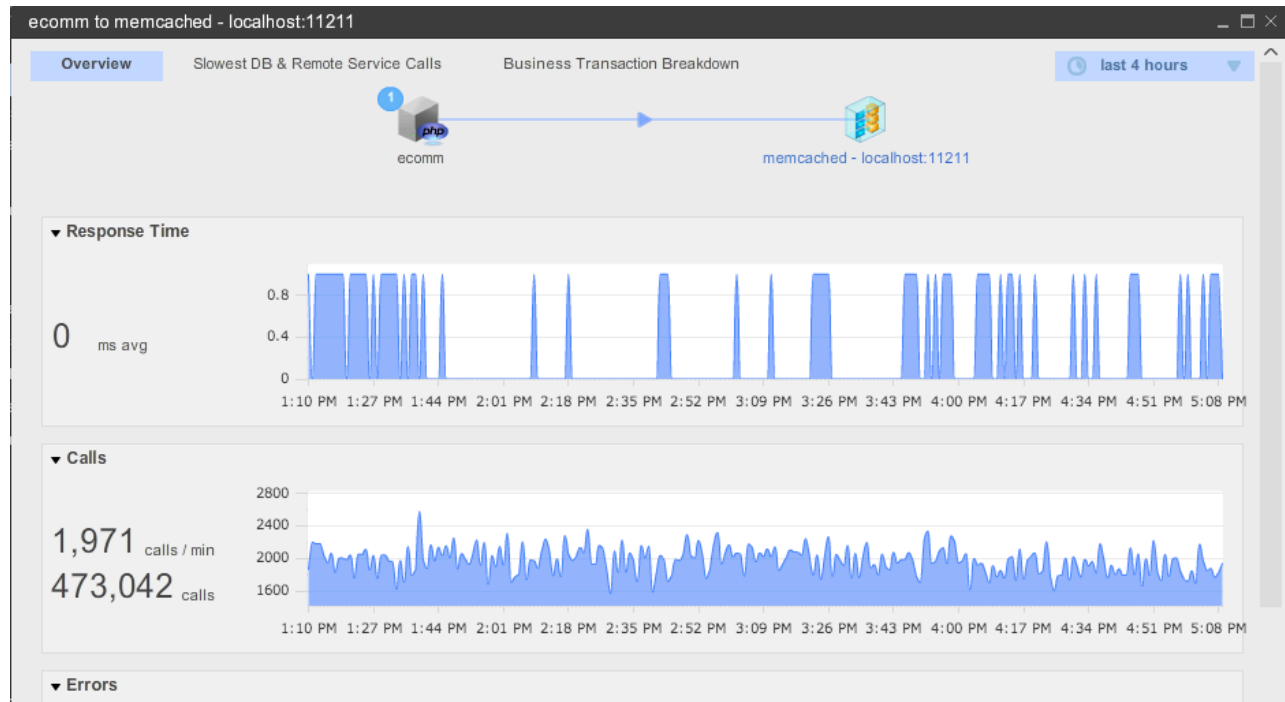
The trend graphs at the bottom of the dashboard show the key performance indicators over the selected time range for the entire application.

## 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.

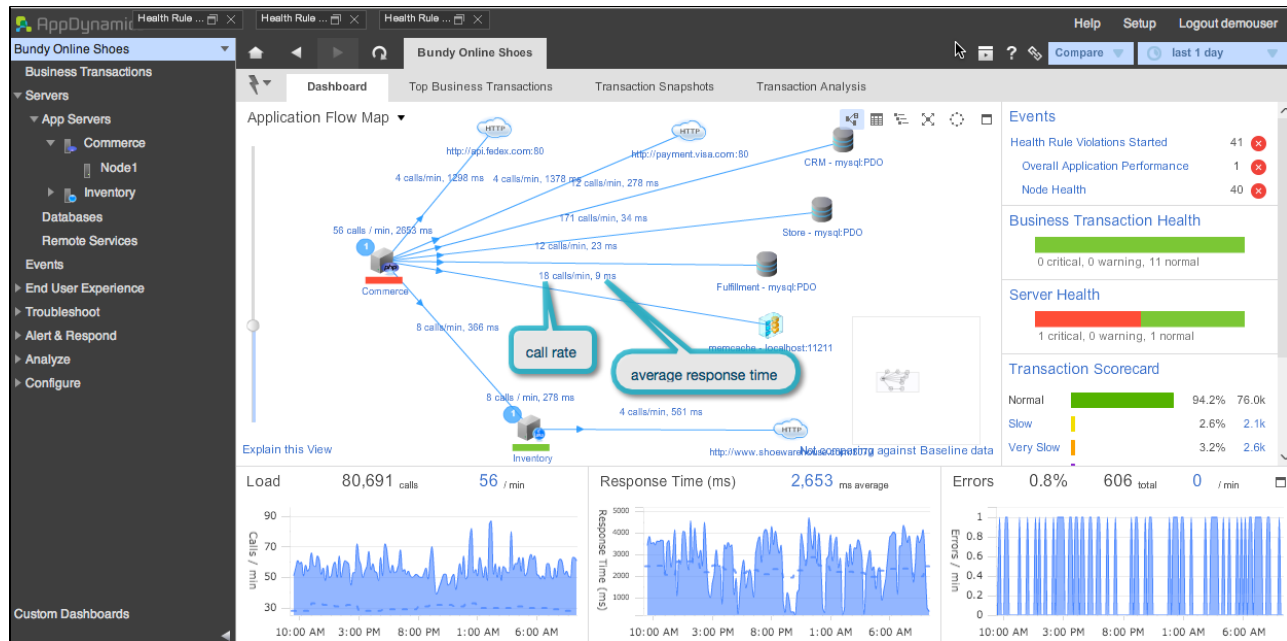
An error is a departure from the expected behavior of a business transaction, which prevents the transaction from working properly.

See [Troubleshoot Errors](#).

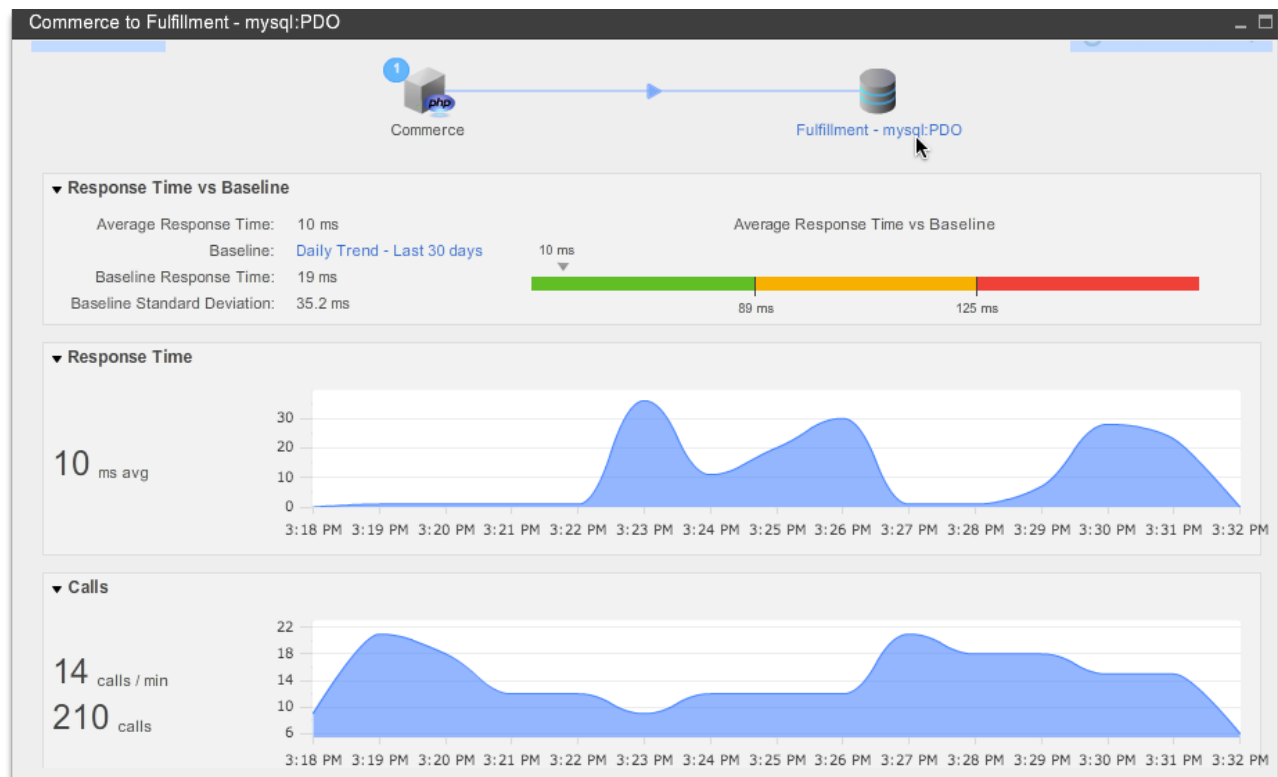
## Tutorial for PHP - Flow Maps

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.

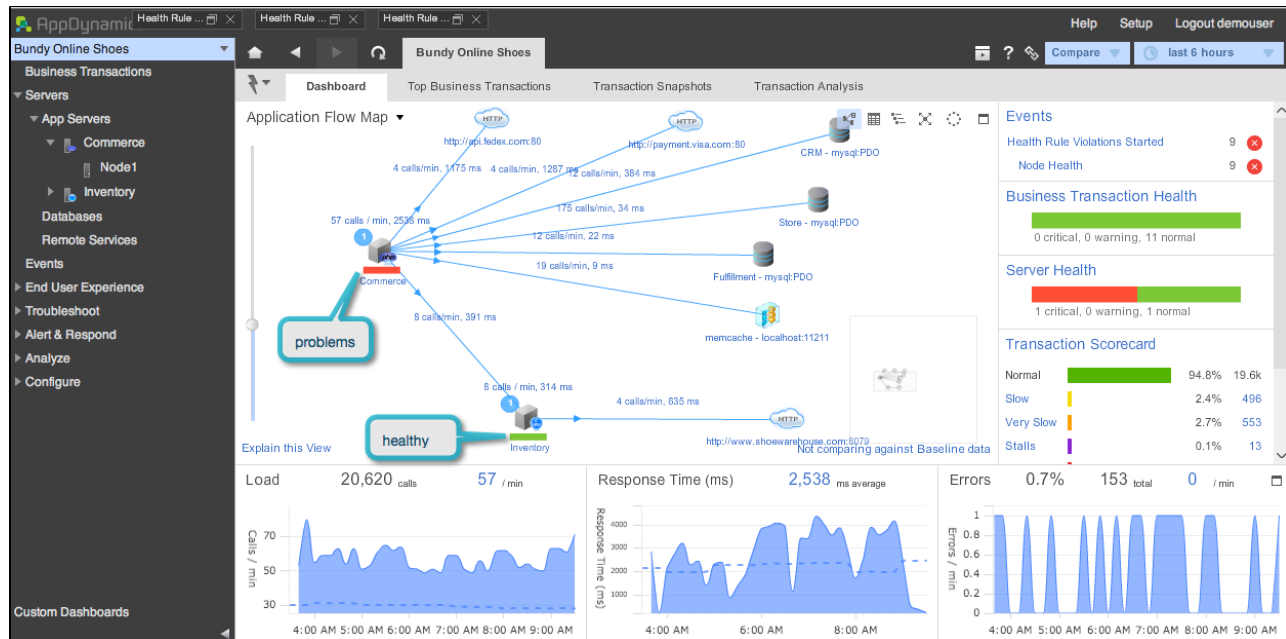
This is the application dashboard with its flow map for a PHP application.



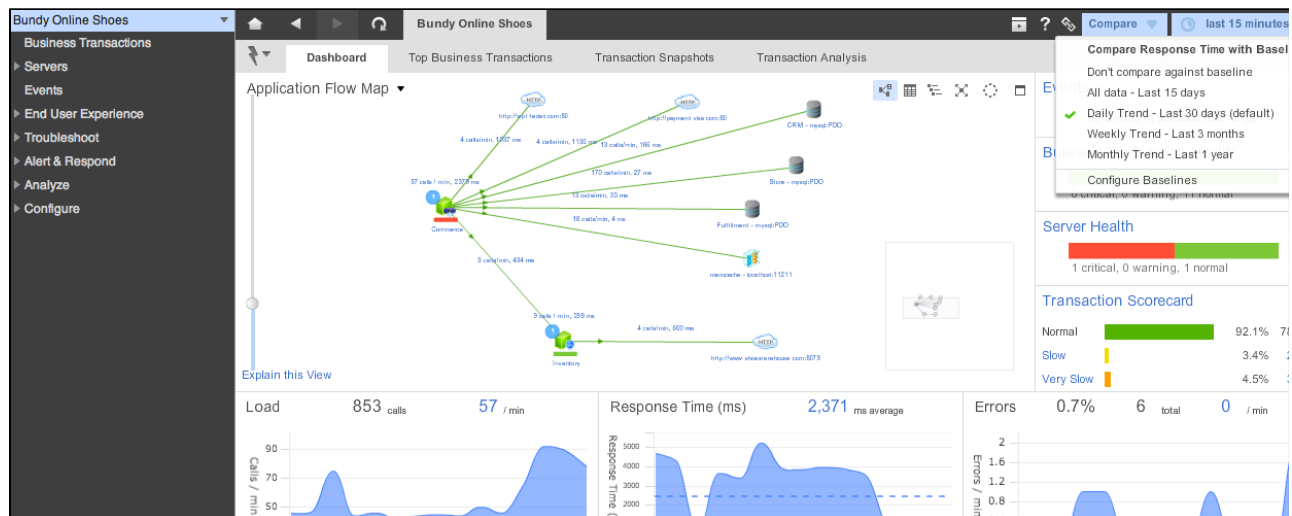
To see a summary of a single flow between two components, click the summary indicators along the line connecting the two components in the flow map. For example, the summary below displays the flow between the Commerce server and the Fulfillment-msq:PDO database.



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:

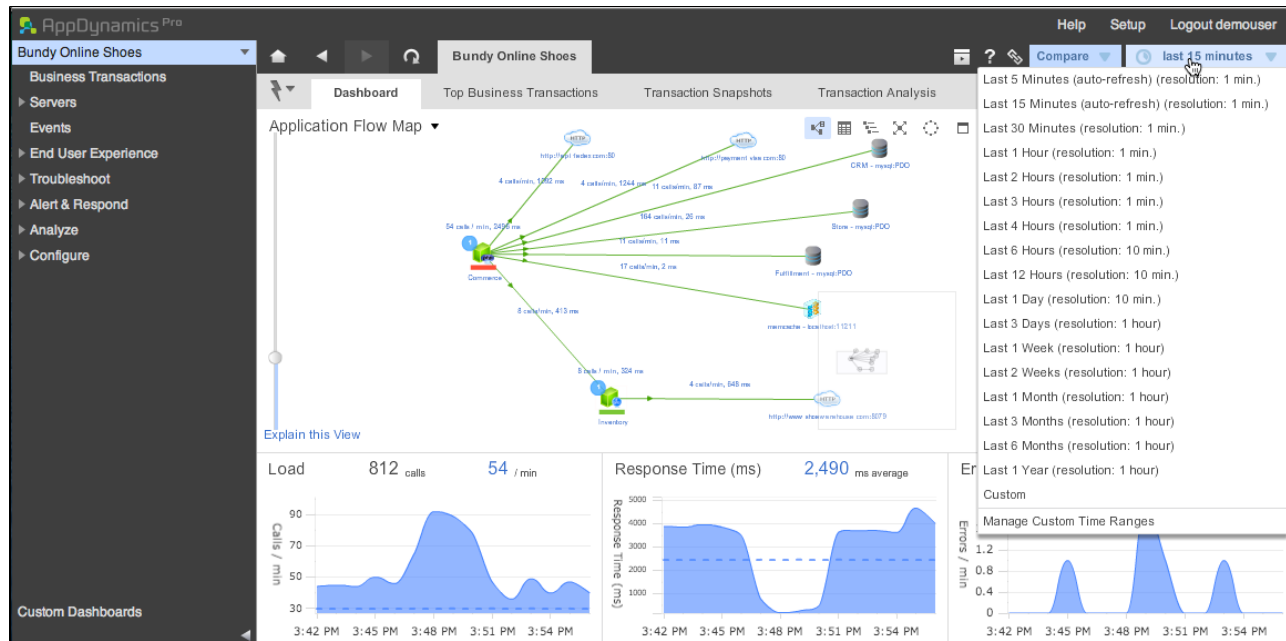


By default, the flow map computes tier health by comparing the state of the tier averaged over the last 15 minutes against the daily trend (the 30 day rolling average). You can change the time window for baseline comparison using the time window pull-down menu. You can also disable baseline comparisons.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

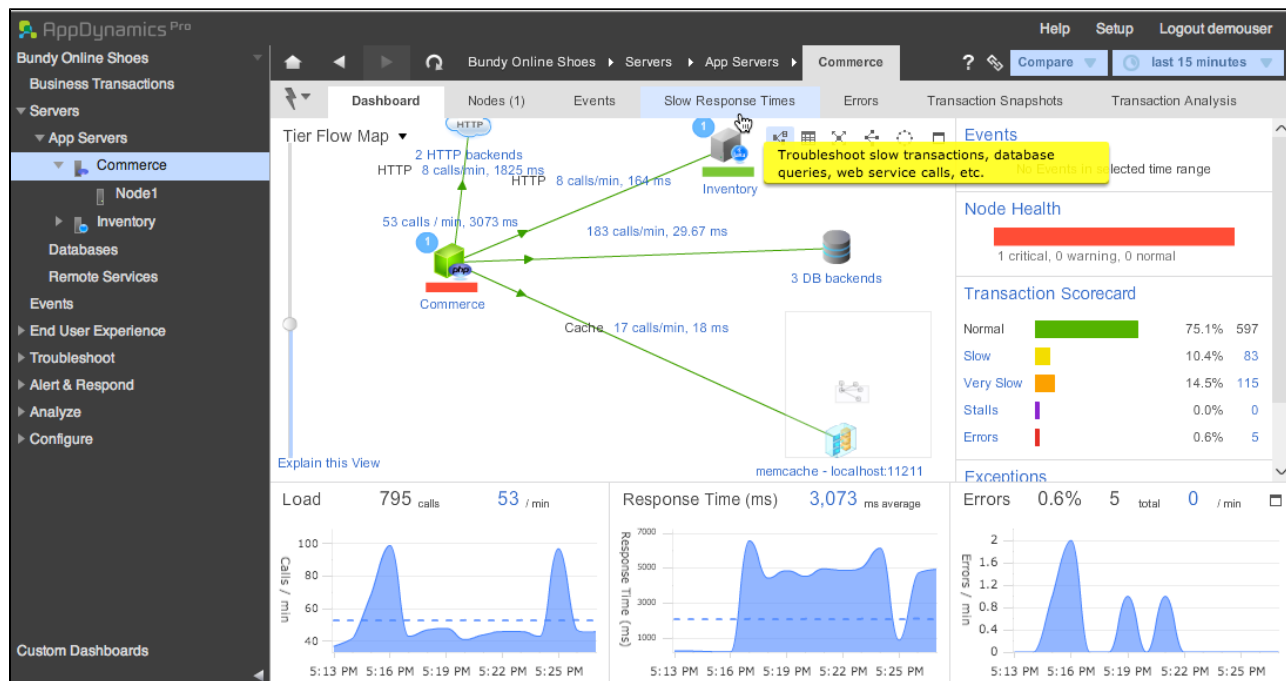


You can change the time range displayed in the flow map using the time window pull down menu. Changing the time range affects all the information in the dashboard and in all other dashboards for the application.

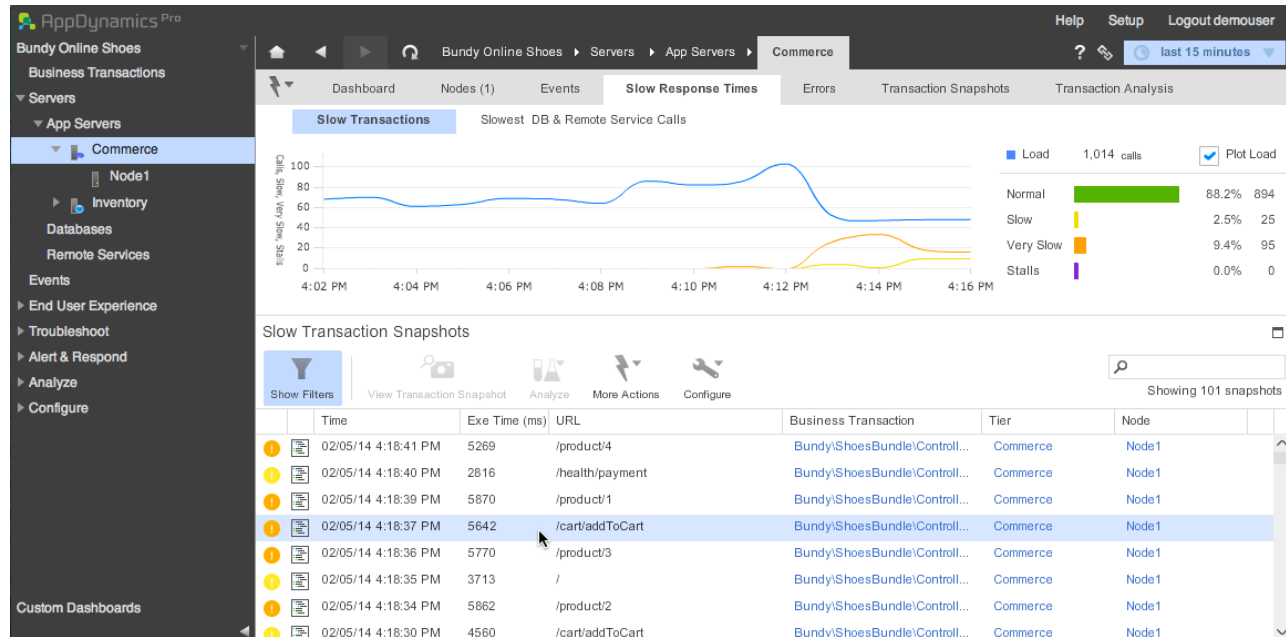




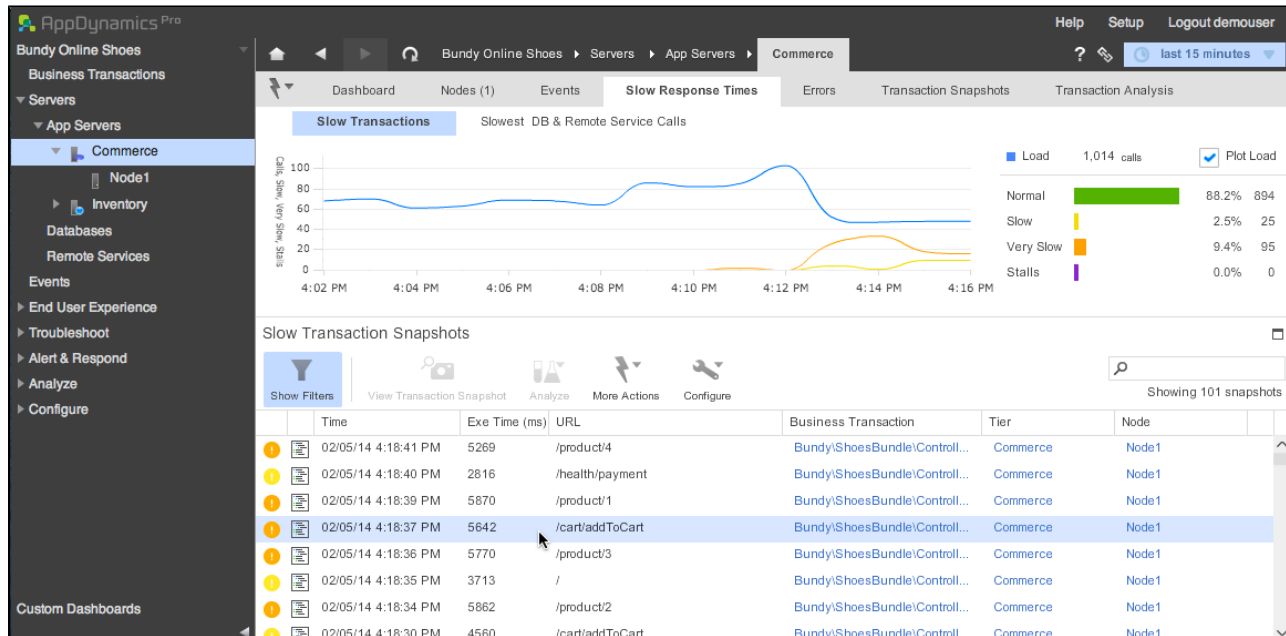
If a tier is experiencing problems, such as slow response times, click the tier name to see a flow map just of that tier:



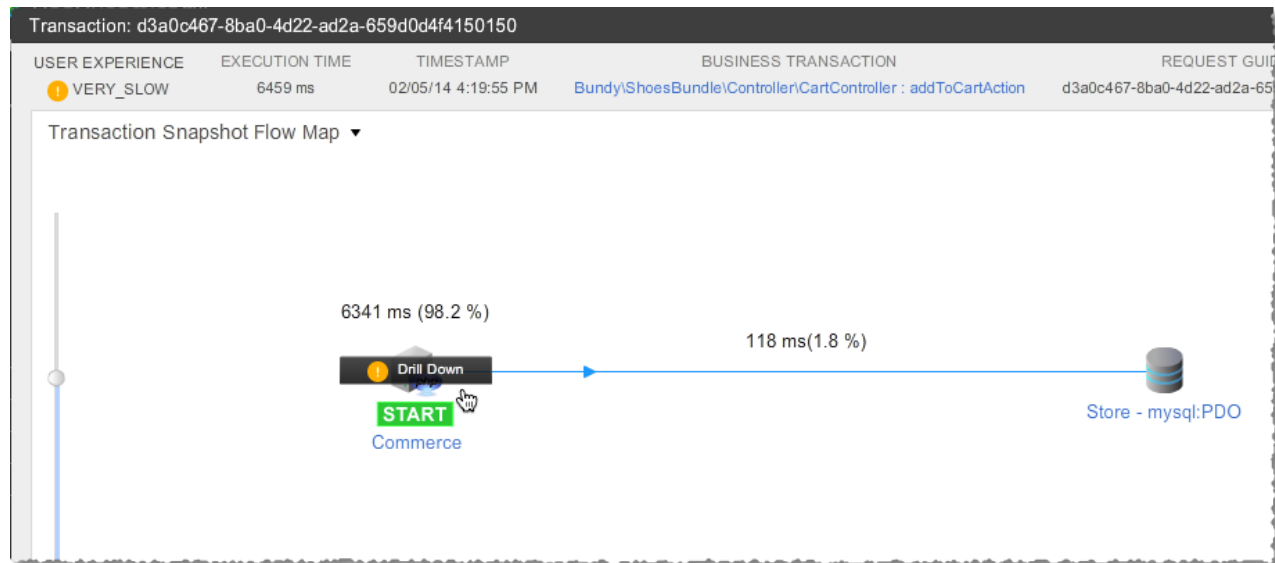
Click the Slow Response Times tab in the tier dashboard to view a graph of the slow response times on that server, optionally compared with the load (blue line), as well as a list snapshots of individual transactions that were slow.



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:



Call Drill Down. Exe Time: 10133 ms. Timestamp: 02/05/14 5:43:30 PM. BT: Bundy\ShoesBundle\Controller\ProductController : viewAction GUID: d3a0c467-8ba0-4d22-ad2a-659d0d4f4150475

Execution Time: 10133 ms. Node Node1. Timestamp: 02/05/14 5:42:41 PM.

Name	Time (ms)
(request) -	0 ms (self) 0 %
(main) - web/app.php	0 ms (self) 0 %
Symfony\Component\HttpKernel\Kernel::handle - web/app.php:27	0 ms (self) 0 %
Symfony\Component\HttpKernel\DependencyInjection\ContainerAwareHttpKernel::handle - app/bootstrap.php.cache:2305	0 ms (self) 0 %
Symfony\Component\HttpKernel\HttpKernel::handle - app/bootstrap.php.cache:3024	0 ms (self) 0 %
Symfony\Component\HttpKernel\HttpKernel::handleRaw - app/bootstrap.php.cache:2885	0 ms (self) 0 %
call_user_func_array - app/bootstrap.php.cache:2913	0 ms (self) 0 %
Bundy\ShoesBundle\Controller\ProductController::viewAction - app/bootstrap.php.cache:2913	112 ms (self) 1.1 %
Doctrine\Common\Persistence\AbstractManagerRegistry::getRepository - Controller/ProductController.php:14	0 ms (self) 0 %
Doctrine\Common\Persistence\AbstractManagerRegistry::getManager - Persistence/AbstractManagerRegistry.php:185	0 ms (self) 0 %
Symfony\Bridge\Doctrine\ManagerRegistry::getService - Persistence/AbstractManagerRegistry.php:185	9172 ms (total) 90.5 %
Doctrine\ORM\EntityRepository::find - Controller/ProductController.php:15	0 ms (self) 0 %
Doctrine\ORM\EntityManager::find - ORM\EntityRepository.php:154	0 ms (self) 0 %
Doctrine\ORM\Persisters\BasicEntityPersister::load - ORM\EntityManager.php:460	849 ms (total) 8.4 %

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

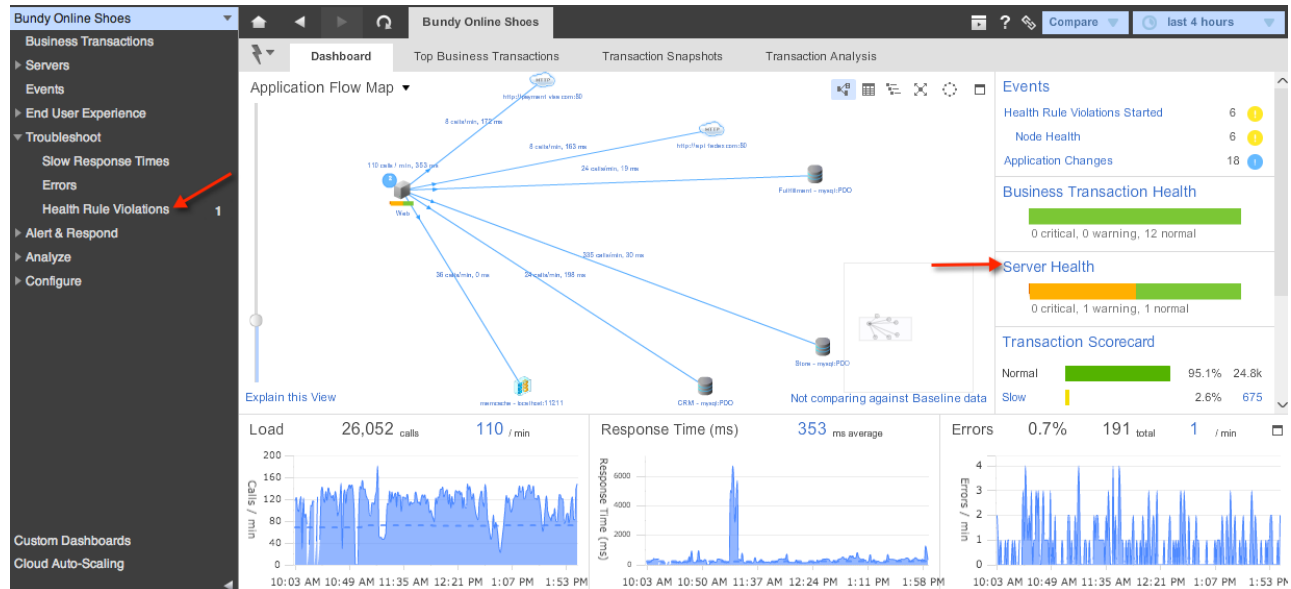
Health is an indicator of how actual performance compares to acceptable performance. Acceptable performance is defined by health rules which generate warning events for performance that may be and critical events for performance that definitely is of concern and requires investigation. A color other 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

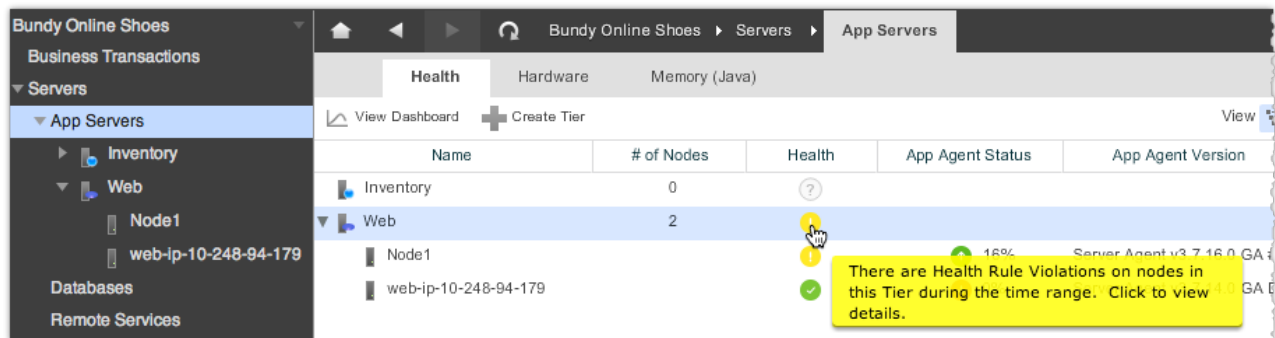
physical memory utilization.

## 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.

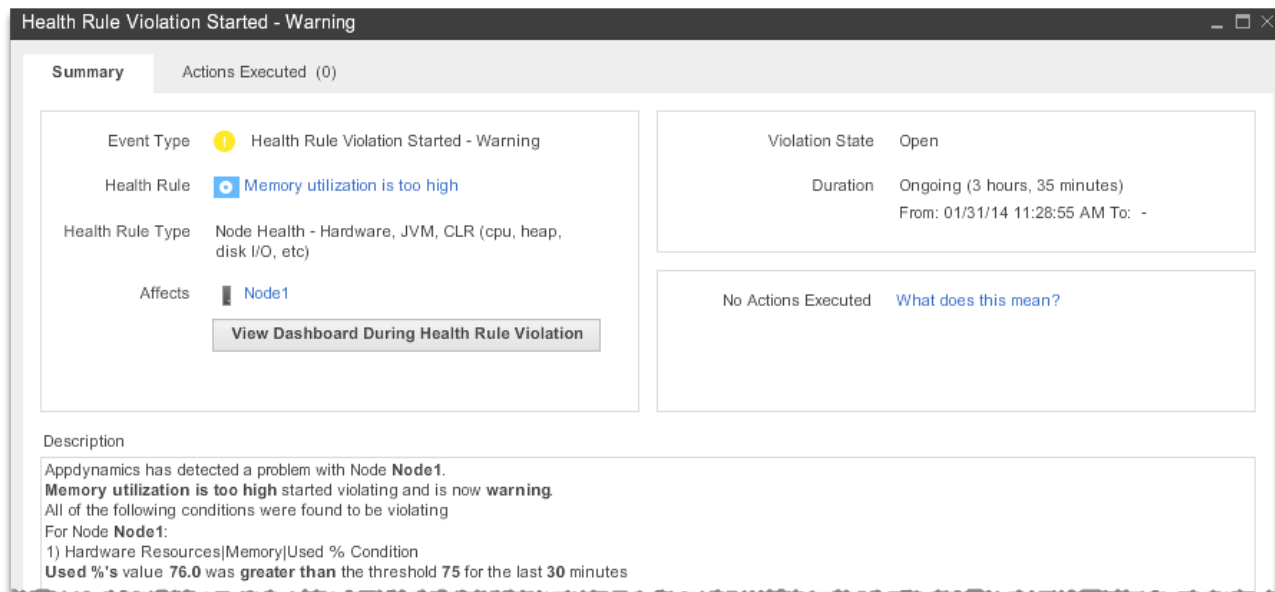


You can view the health rule violation details and the status of the violation:




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](#)

[View Dashboard During Health Rule Violation](#)

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%.

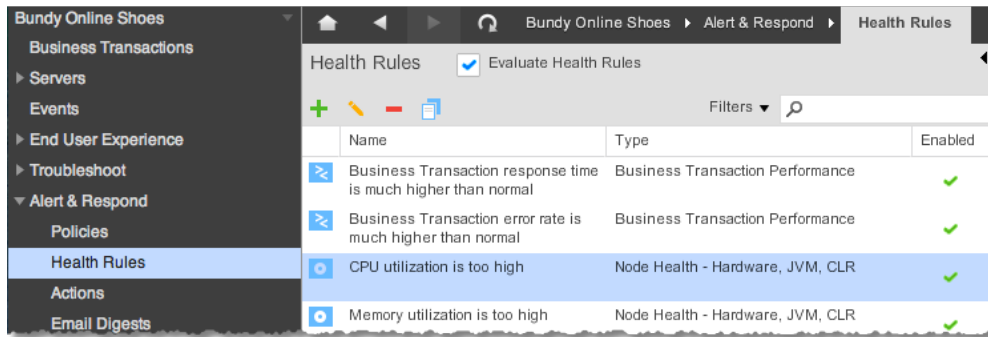
To access the node health rule configuration window

Do one of the following:

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

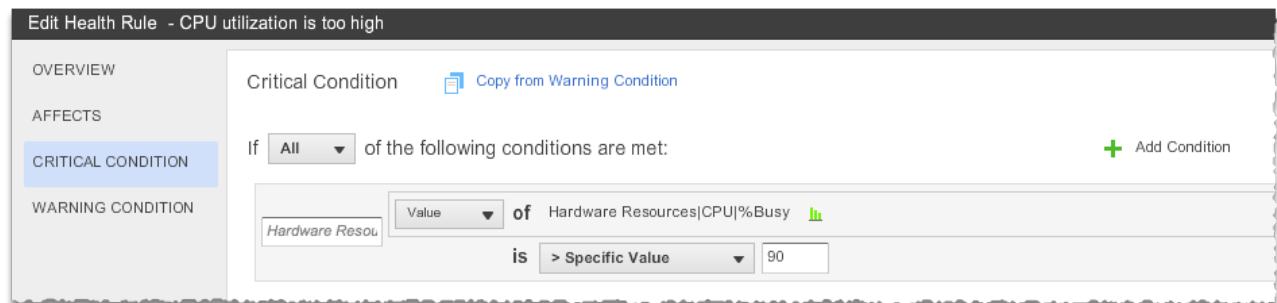


- In the left navigation pane:
  - Click **Alert & Respond->Health Rules**.
  - Double-click the the health rule in the health rule list.



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**.



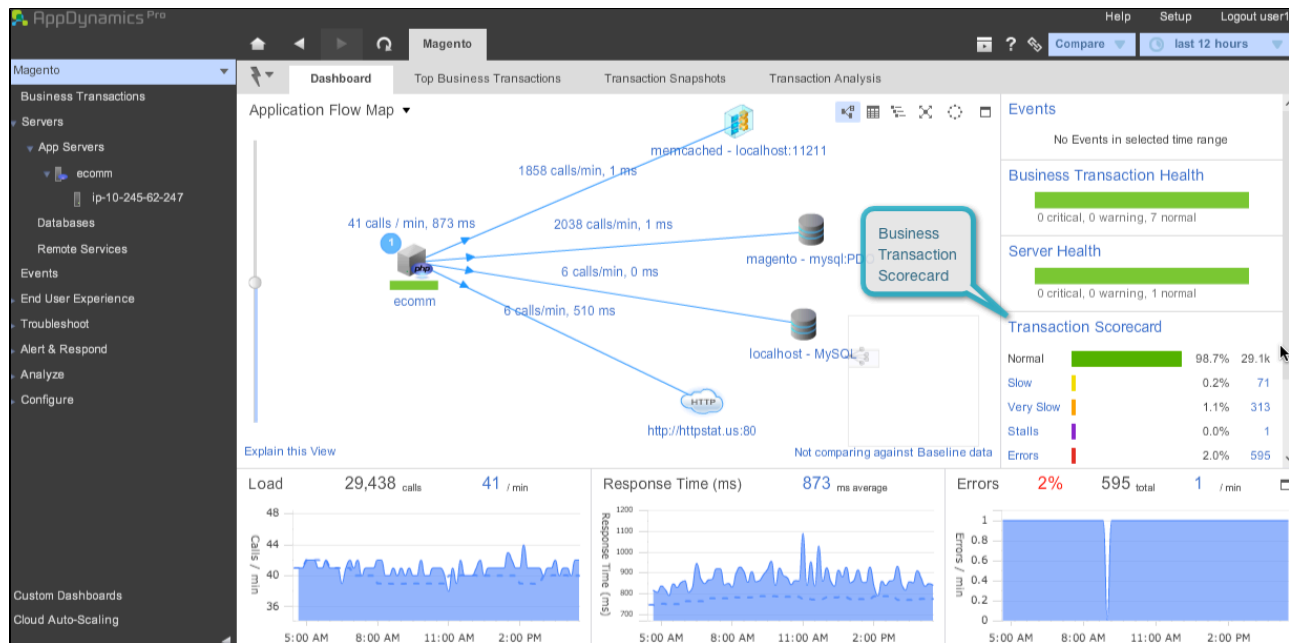
You can also add more conditions required to trigger the health rule violation by clicking **Add Condition**. Or change the times at which the health rule is in effect, in the Overview tab. Or restrict the nodes and tiers affected by the health rule in the Affects tab. Or create entirely new health rules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

## Learn More

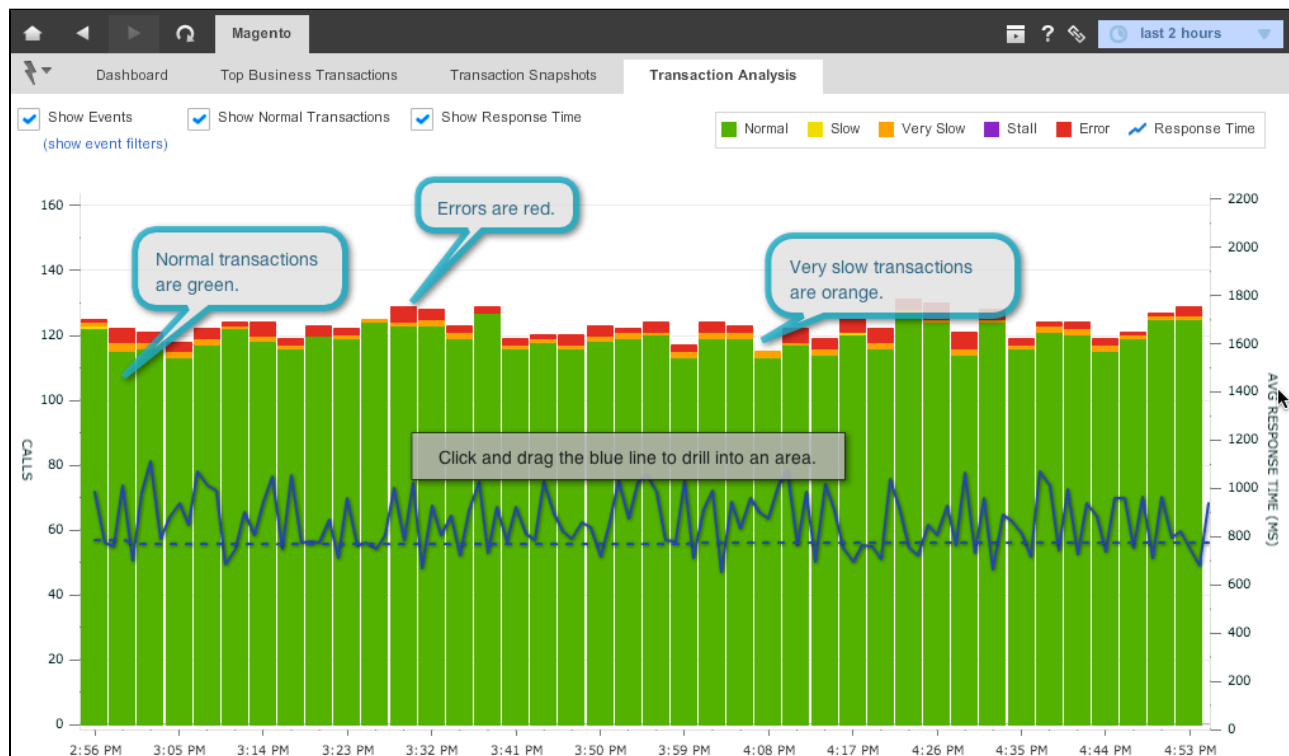
- [Health Rules](#)
- [Configure Health Rules](#)
- [Policies](#)
- [Actions](#)
- [Troubleshooting Server Health, AppDynamics in Action video](#) 

## 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.

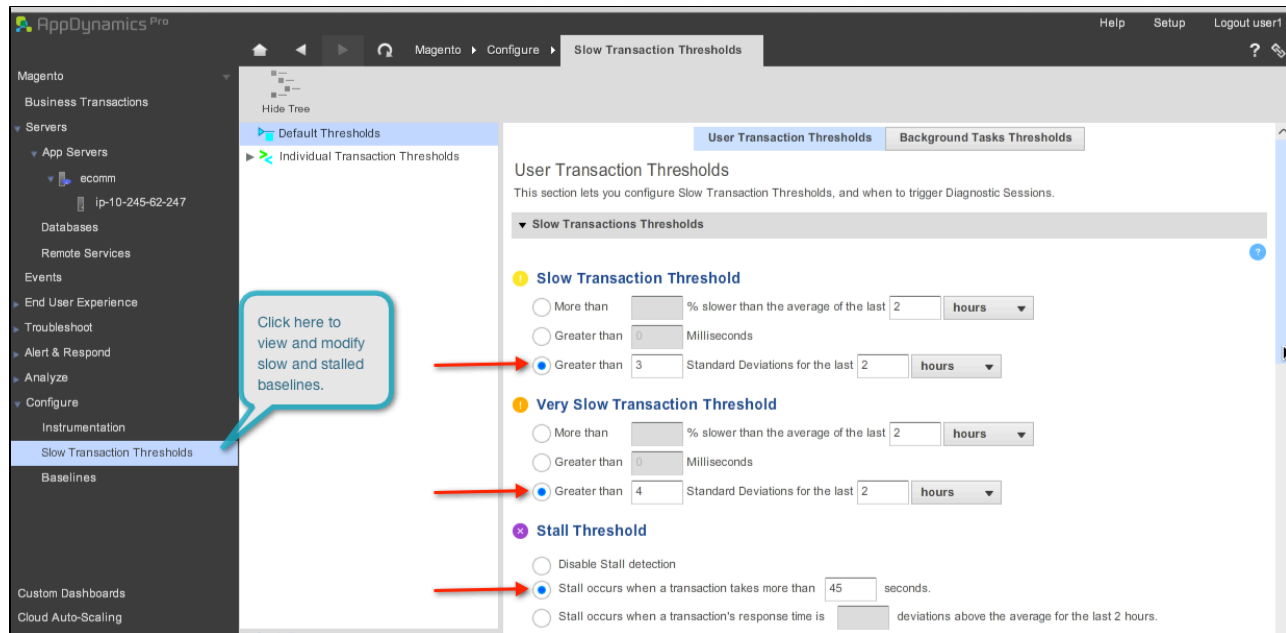


To see the response time distribution over time, click the Transaction Scorecard link or the Transaction Analysis tab in a dashboard. For more information about this histogram see [Transaction Analysis Tab](#).

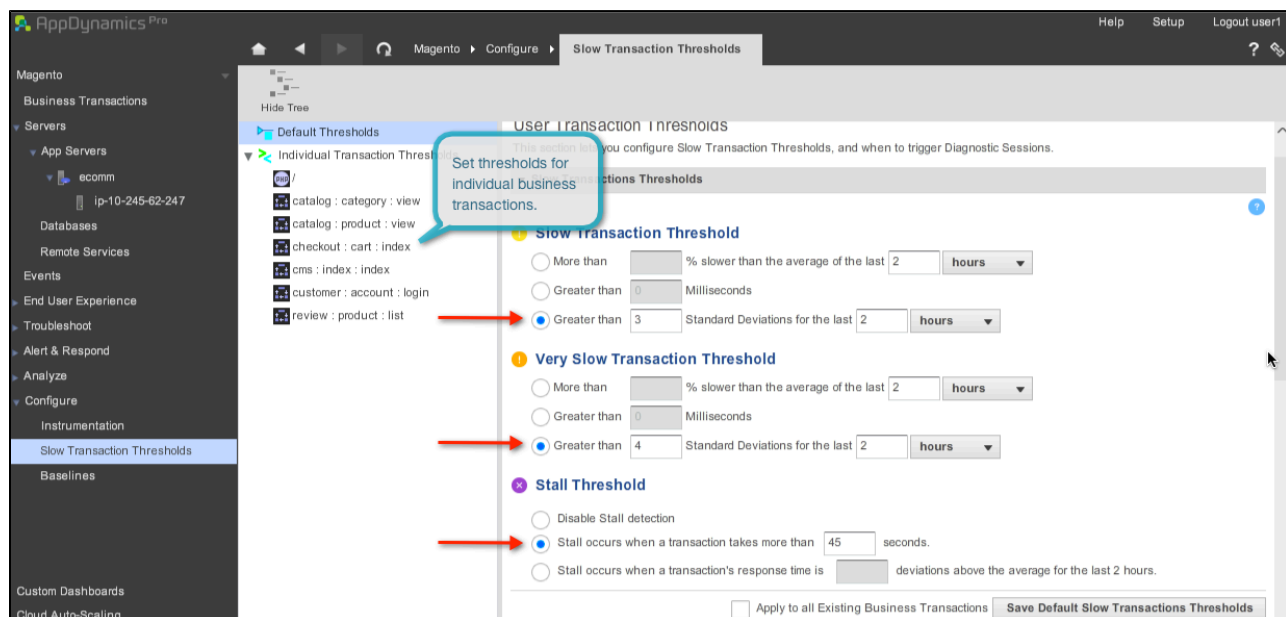


To view or modify the default thresholds, click **Configure->Slow Transaction Thresholds** in the left navigation pane.





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](#).



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



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

## Administer App Agents for PHP

### App Agent for PHP Proxy Configuration Properties

- [Where to Configure App Agent Properties](#)
- [Example PHP Proxy controller-info.xml File](#)
- [Proxy Properties](#)
  - [Proxy-Controller Communication Properties](#)
    - [Controller Host Property](#)
    - [Controller Port Property](#)
  - [Proxy Identification Properties](#)
    - [Application Name Property](#)
    - [Tier Name Property](#)
    - [Node Name Property](#)
  - [Multit-Tenant Mode Properties](#)
    - [Account Name Property](#)
    - [Account Access Key Property](#)
  - [Other Properties](#)
    - [Controller SSL Enabled Property](#)
    - [Enable Orchestration Property](#)
    - [Force Agent Registration Property](#)
- [Learn More](#)

These properties support communication between the PHP proxy component and the Controller.

### Where to Configure App Agent Properties

You can configure the PHP proxy properties in the controller-info.xml file in the <php\_agent\_install>/proxy/conf directory.

You can also set most of these properties on the command-line when you install the agent. See [Install the App Agent for PHP](#).

The installer generates a new controller-info.xml file every time it runs. You can edit controller-info.xml to add, delete or modify the proxy configuration properties, but be aware that your changes are overwritten every time you re-install the agent, so if you reinstall the agent you need to update controller-info.xml manually.

The only properties that cannot be added on the install command-line are [Enable Orchestration Property](#) and [Force Agent Registration Property](#).

### Example PHP Proxy controller-info.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<controller-info>

  <controller-host>192.168.1.20</controller-host>

  <controller-port>8090</controller-port>

  <controller-ssl-enabled>>false</controller-ssl-enabled>

  <application-name>Magento</application-name>

  <tier-name>php</tier-name>

  <node-name>php0</node-name>

  <controller-ssl-enabled>>true</controller-ssl-enabled>

  <enable-orchestration>>false</enable-orchestration>

</controller-info>
```

## Proxy Properties

This section describes the proxy configuration properties.

### Proxy-Controller Communication Properties

#### Controller Host Property

**Description:** This is the host name or the IP address of the AppDynamics Controller. Example values are 192.168.1.22 or myhost or myhost.abc.com. This is the same host that you use to access the AppDynamics browser-based user interface. For an on-premise Controller, use the value for application server host name that was configured when the Controller was installed.

Set when invoking the agent installer.

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

**Type:** String

**Default:** None

**Required:** Yes

#### Controller Port Property

**Description:** This is the HTTP(S) port of the AppDynamics Controller. This is the same port that you use to access the AppDynamics browser-based user interface.

If the Controller SSL Enabled property is set to true, specify the HTTPS port of the Controller; otherwise specify the HTTP port. See [Controller SSL Enabled Property](#).

Set when invoking the agent installer.

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

**Type:** Positive Integer

**Default:** For on-premise installations, port 8090 for HTTP and port 8181 for HTTPS are the defaults.

**Required:** Yes

### Proxy Identification Properties

#### Application Name Property

**Description:** This is the name of the logical business application that the instrumented node belongs to.

If a business application of the configured name does not exist, it is created automatically.

Set when invoking the agent installer.

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

**Type:** String

**Default:** None

**Required:** Yes

#### Tier Name Property

**Description:** This is the name of the logical tier that this node belongs to.

Set when invoking the agent installer.

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

**Type:** String

**Default:** None

**Required:** Yes

#### Node Name Property

**Description:** This is the name of the instrumented node.

In general, the node name must be unique within the business application and physical host.

Set when invoking the agent installer.

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

**Type:** String

**Default:** None

**Required:** Yes

### Multitenant Mode Properties

**Description:** If the AppDynamics Controller is running in multi-tenant mode or if you are using the AppDynamics SaaS Controller, specify the account name and account access key for this agent to use to authenticate with the Controller. If you are using the AppDynamics SaaS Controller, the Welcome email sent by AppDynamics provides this information.

If the Controller is running in single-tenant mode (the default) there is no need to configure these values.

## Account Name Property

**Description:** This is the account name used to authenticate with the Controller.

Can be set when invoking the agent installer.

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

**Type:** String

**Default:** None

**Required:** Yes for AppDynamics SaaS Controller and other multi-tenant users; no for single-tenant users.

## Account Access Key Property

**Description:** This is the account access key used to authenticate with the Controller.

Can be set when invoking the agent installer.

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

**Type:** String

**Default:** None

**Required:** Yes for AppDynamics SaaS Controller and other multi-tenant users; no for single-tenant users.

## Other Properties

### Controller SSL Enabled Property

**Description:** When set to true, this property specifies that the agent should use SSL (HTTPS) to connect to the Controller. If SSL Enabled is true, set the Controller Port property to the HTTPS port of the Controller. See [Controller Port Property](#).

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

**Type:** Boolean

**Default:** False

**Required:** No

### Enable Orchestration Property

**Description:** When set to true, enables auto-detection of the controller host and port when the app server is a compute cloud instance created by an AppDynamics orchestration workflow. See

### Controller Host Property and Controller Port Property.

In a cloud compute environment, auto-detection is necessary for the Create Machine tasks in the workflow to run correctly.

If the host machine on which this agent resides is not created through AppDynamics workflow orchestration, this property should be set to false.

**Element in controller-info.xml:** <enable-orchestration>

**Type:** Boolean

**Default:** False

### Force Agent Registration Property

**Description:** Set to true only under the following conditions:

- The Agent has been moved to a new application and/or tier from the UI and
- You want to override that move by specifying a new application name and/or tier name in the agent configuration.

**Element in controller-info.xml:** <force-agent-registration>

**Type:** Boolean

**Default:** False

**Required:** No

### Learn More

- [Install the App Agent for PHP](#)
- [Logical Model](#)