

# AppDynamics Pro Documentation

## AppDynamics Essentials



1. AppDynamics Essentials	2
1.1 Features Overview	2
1.2 Architecture	5
1.3 Logical Model	7
1.3.1 Hierarchical Configuration Model	8
1.3.2 Mapping Application Services to the AppDynamics Model	9
1.4 Getting Started	11
1.4.1 Get Started with AppDynamics SaaS	11
1.4.1.1 Use a SaaS Controller	14
1.4.1.2 SaaS Availability and Security	15
1.4.2 Get Started With AppDynamics On-Premise	16
1.4.3 Download AppDynamics Software	19
1.4.4 Quick Start for DevOps	20
1.4.5 Quick Start for Architects	21
1.4.6 Quick Start for Administrators	22
1.4.7 Quick Start for Operators	23
1.4.8 Set User Preferences	23
1.5 Glossary	25
1.6 AppDynamics Support	30
1.6.1 Controller Dump Files	31
1.6.2 Log Files for Troubleshooting	32
1.6.3 Configure Remote Monitoring of an On-Premise Controller	32
1.6.4 Download Doc PDFs	35
1.6.5 Use the Documentation Wiki	35
1.6.6 License Information	37
1.7 Documentation Map	38

# AppDynamics Essentials

## AppMan Advice



*Isn't it about time you mapped your app?*

AppDynamics provides application performance management for modern application architectures. Designed for distributed SOA environments, AppDynamics helps you to manage service levels, reduce mean-time-to-repair for problems, plan for application efficiency, and automate typical life-cycle actions for distributed applications.

## Basics

[Features Overview](#)  
[AppDynamics in Action Videos](#)  
[Architecture](#)  
[Logical Model](#)  
[Glossary](#)

## Getting Started

[Get Started with AppDynamics SaaS](#)  
[Get Started With AppDynamics On-Premise](#)  
[Download AppDynamics Software](#)  
[Set User Preferences](#)

[Quick Start for Operators](#)  
[Quick Start for DevOps](#)  
[Quick Start for Architects](#)  
[Quick Start for Administrators](#)

## Support

[Release Notes](#)  
[Supported Environments](#)

[AppDynamics Support](#)  
[Documentation Map](#)

## Features Overview

- [Continuous Discovery, Visibility, and Problem Detection](#)

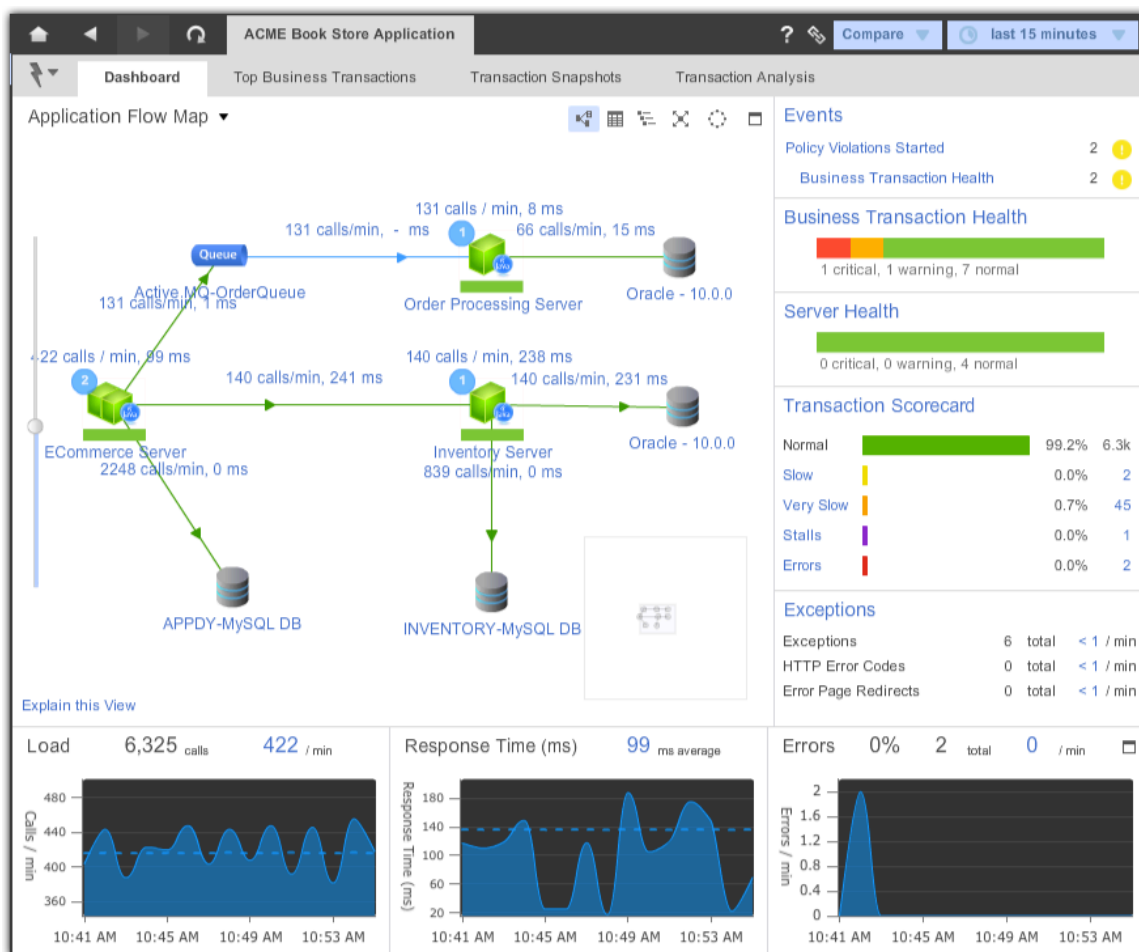
- Real-Time Business Transaction Monitoring
- End User Monitoring
- Hardware and Server Monitoring
- Health Rules, Policies, and Actions
- Troubleshooting and Diagnostics
- Systems Integration
- [Learn More](#)

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

## Continuous Discovery, Visibility, and Problem Detection

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

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

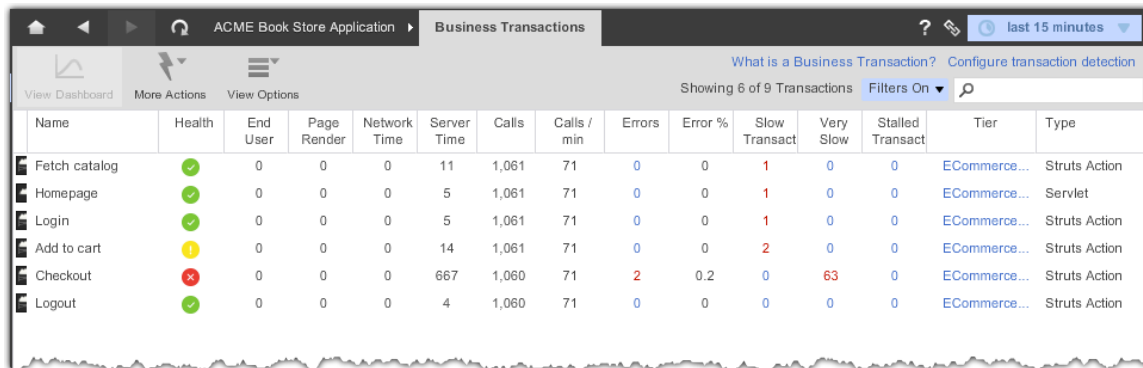


AppDynamics observes normal performance patterns so that it knows when application performance becomes abnormal. It automatically identifies metrics whose current values are out of the normal range, based on dynamic baselines it has observed for these metrics. See [Behavior Learning and Anomaly Detection](#).

## Real-Time Business Transaction Monitoring

An AppDynamics business transaction represents a distinct logical user activity such as logging in, searching for items, buying an item, etc. Organizing application traffic into business transactions aligns the traffic with the primary functions of a web business. This

approach focuses on how your users are experiencing the site and provides real-time performance monitoring.



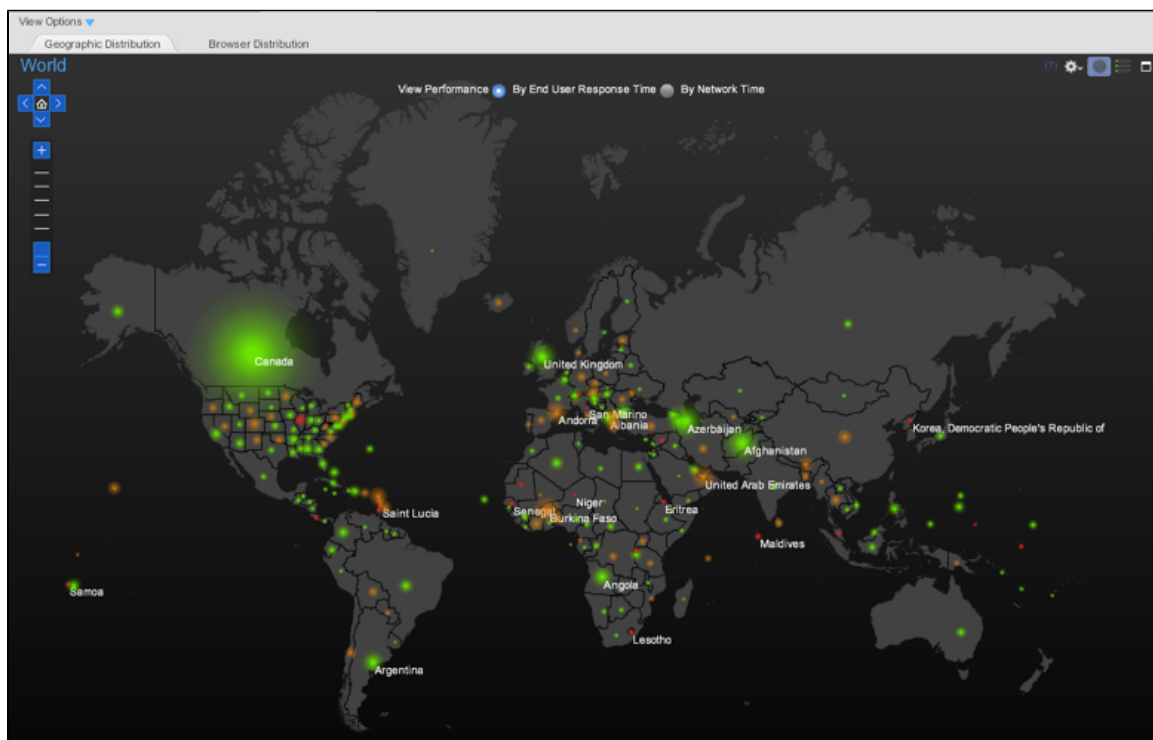
Name	Health	End User	Page Render	Network Time	Server Time	Calls	Calls / min	Errors	Error %	Slow Transact	Very Slow	Stalled Transact	Tier	Type
Fetch catalog	✓	0	0	0	11	1,061	71	0	0	1	0	0	ECommerce...	Struts Action
Homepage	✓	0	0	0	5	1,061	71	0	0	1	0	0	ECommerce...	Servlet
Login	✓	0	0	0	5	1,061	71	0	0	1	0	0	ECommerce...	Struts Action
Add to cart	⚠	0	0	0	14	1,061	71	0	0	2	0	0	ECommerce...	Struts Action
Checkout	✗	0	0	0	667	1,060	71	2	0.2	0	63	0	ECommerce...	Struts Action
Logout	✓	0	0	0	4	1,060	71	0	0	0	0	0	ECommerce...	Struts Action

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

## End User Monitoring

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

- Where are the heaviest loads?
- Where are the slowest end-user response times?
- How does end user performance vary by Web browser?



See [End User Experience](#).

## Hardware and Server Monitoring

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

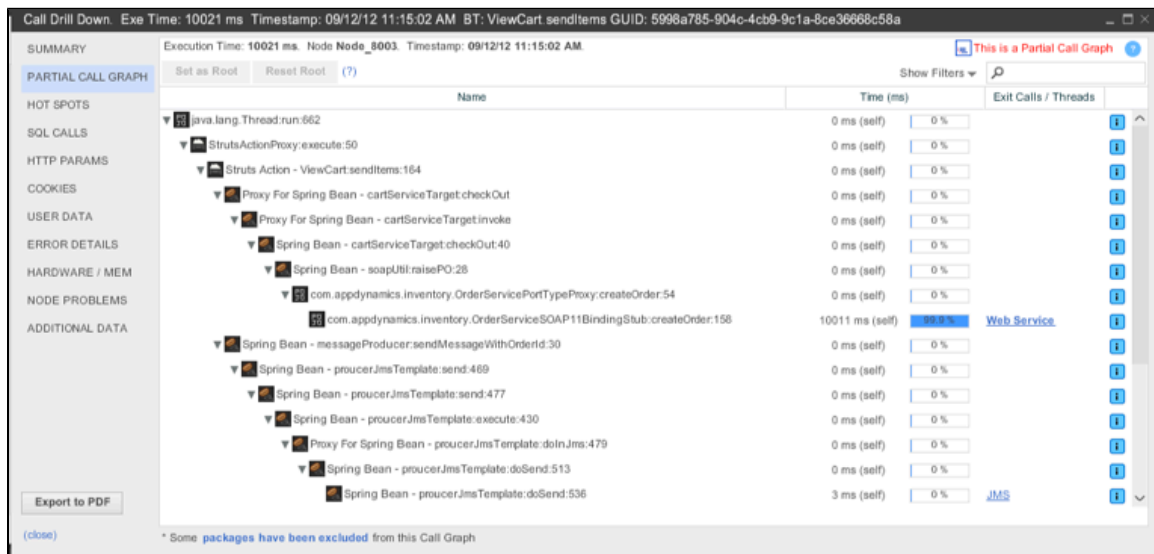
## Health Rules, Policies, and Actions

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

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

## Troubleshooting and Diagnostics

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



See [Rapid Troubleshooting](#).

## Systems Integration

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

## Learn More

- [Product Features and Benefits](#)

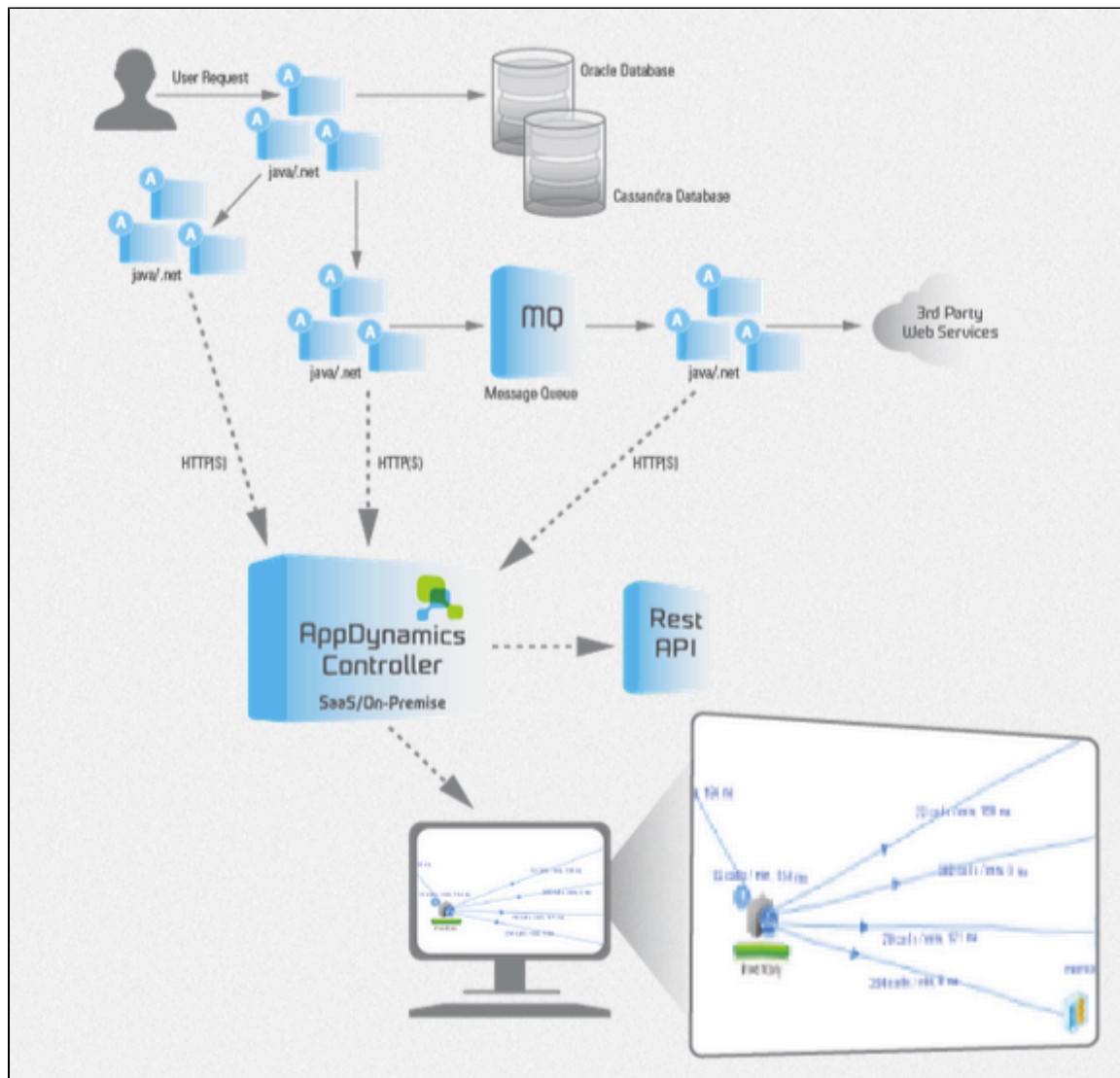
## Architecture

- [AppDynamics Pro Architecture](#)
  - [AppDynamics Controller and UI](#)
  - [AppDynamics App Agents](#)
  - [AppDynamics Machine Agents](#)
- [Learn More](#)

This topic summarizes the components of AppDynamics and how they work together to monitor your application environment.

## AppDynamics Pro Architecture

An AppDynamics deployment consists of a Controller (either on-premise or SaaS) and its UI, app agents, and machine agents.



## AppDynamics Controller and UI

The AppDynamics Controller is the central repository and analytics engine where all performance data is stored, baselined, and analyzed. The Controller is specially designed for large-scale production environments, and can scale to manage hundreds to thousands of application servers.

The AppDynamics Controller can be installed on-premise or it can be accessed as software as a service (SaaS). A SaaS Controller is managed at AppDynamics and you connect to it from a web browser using HTTP/HTTPS. An on-premise Controller is managed by you on your server in a data center or in the cloud.

You access performance data interactively using the Controller UI or programmatically using the AppDynamics REST API.

## AppDynamics App Agents

AppDynamics app agents are installed on your JVM or .NET application. They automatically inject instrumentation in application bytecode at runtime.

Patent-pending Dynamic Flow Mapping (TM) technology continuously discovers, maps, and tracks all business transactions, services, and backends in your web application architecture 24x7.

Patent-pending Deep-on-Demand Diagnostics (TM) technology learns code execution behavior for each business transaction. It automatically detects problems and collects deep diagnostics data to troubleshoot them.

## AppDynamics Machine Agents

One or more machines (real or virtual) constitute the hardware and operating system on which your application runs. Machines can be instrumented by an AppDynamics machine agent, which collects data about machine performance and sends it to the Controller.

## Learn More

- [Logical Model](#)
- [AppDynamics Administration](#)

## Logical Model

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

This topic describes the basic elements of the AppDynamics model.

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

## Business Application

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

A business application usually does not map directly to only one Java or .NET application, and often it maps to more than one. See [Mapping Application Services to the AppDynamics Model](#).

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

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

Business applications contain tiers, and tiers contain nodes.

## Tiers

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

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

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

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

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

## Nodes

A node is the basic unit of processing that AppDynamics monitors. By definition a node is instrumented by an AppDynamics agent, either an app agent or machine agent or both. App agents are installed on JVMs and Windows .NET applications (IIS, executables, or services). Machine agents are installed on virtual or physical machine operating systems.

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

## Learn More

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

## Hierarchical Configuration Model

- [Entry Point and Exit Point Inheritance](#)
- [Node Inheritance](#)
- [Switching Configuration Levels](#)
- [Learn More](#)

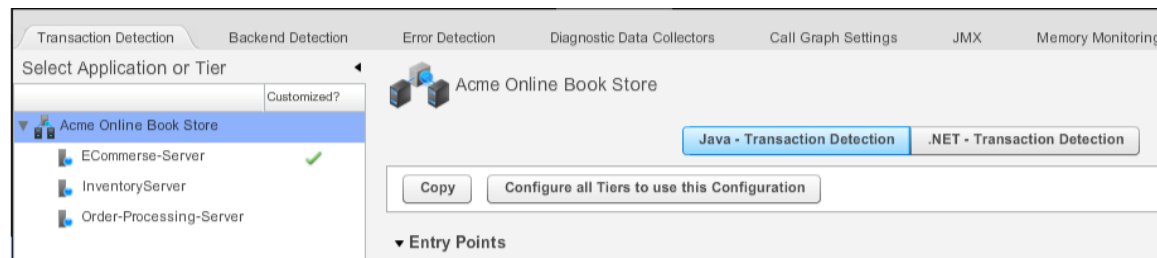
Transaction detection (entry point), backend detection (exit point), and node property configurations are applied on a hierarchical inheritance model. This model provides a default configuration for new tiers as well as the ability to re-use custom configurations in all tiers or tiers that you specify,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

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

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

## Entry Point and Exit Point Inheritance

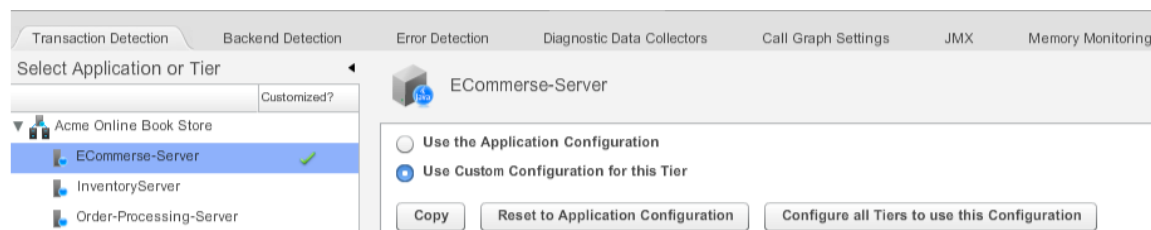
By default, tiers inherit the entry point and exit point configurations of the application. You can copy the application-level configuration to specific tiers or explicitly configure all tiers to use the application-level configuration.



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

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

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

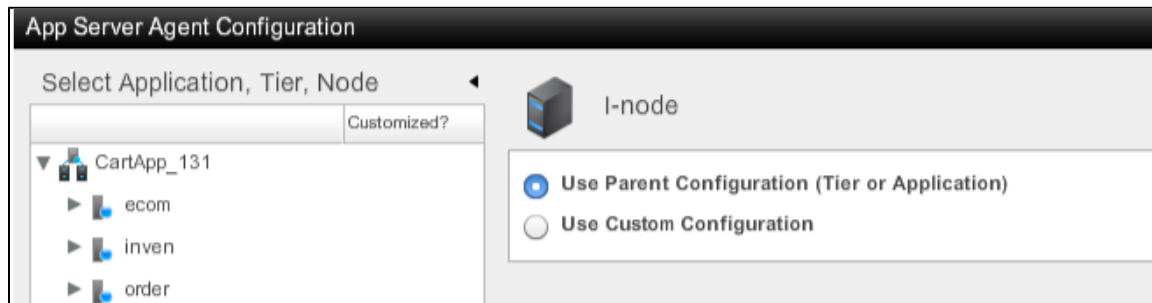




## Node Inheritance

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

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



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

## Switching Configuration Levels

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

## Learn More

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

## Mapping Application Services to the AppDynamics Model

- [Your Application and the AppDynamics Model](#)
- [The AppDynamics Business Application](#)
  - [One or More Business Applications](#)
  - [Tier](#)
  - [Node](#)
- [Naming Guidelines](#)
  - [Learn More](#)

This topic discusses how to map your application environment's services to AppDynamics business applications, tiers, and nodes. For an overview see [Logical Model](#).

Your team may use different terminology to describe your application environment such as "site", "system" or "managed environment". You may have other terms for the services, such as "resource", "module" or "component". For the purposes of this discussion the terms "application environment" and "services" are used.

## Your Application and the AppDynamics Model

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

- Web applications served from a JVM, IIS or other platform
- Databases or other data stores
- Services such as message queues and caches

In a large organization, different business units may be responsible for different services. When you finish the model and configure AppDynamics, you can also create specific alerts and dashboards for the different service owners.

Design how your services are mapped and modeled in AppDynamics so that the flow map shows:

- All relevant tiers
- All relevant databases and remote services
- The business transactions flowing through the tiers

AppDynamics models your application environment into a hierarchical set of business applications, tiers, and nodes. The hierarchy is important: a node cannot belong to more than one tier, and a tier cannot belong to more than one business application.

Once you understand the model and configure the AppDynamics business application, tiers, and nodes, you can monitor your application traffic through all its services to rapidly identify existing and potential problems.

## The AppDynamics Business Application

An AppDynamics business application models all of your application environment's services that provide a complete set of functionality. Think of it as all the web applications, databases, and services that interact or "talk" to each other or to a shared service. When web applications, databases, and services interact, AppDynamics can correlate their activities to provide useful and interesting performance data. If services exchange information, then they should be monitored in the same business application. You would give them the same business application name when you configure AppDynamics agent properties during installation.

For example, a business application maps to a shopping application environment that is composed of an inventory application, e-commerce frontend application, and databases. All of these services interact with one or more of the others.

## One or More Business Applications

There are no hard-and-fast rules for deciding what you want to monitor as a business application for your application environment. Each environment has its own needs. Here are some guidelines:

- AppDynamics lets you monitor multiple business applications; however activities cannot be correlated between business applications. If you want events to correlate make sure the services are in the same business application.
- If all of your services interact, then you can have one big business application.
- If a service doesn't interact with any others, then it can have its own business application.
- If there are multiple sets of services that can be grouped, you can have multiple business applications.

A good example of using multiple business applications is when you have staging, testing, and production environments for the same website. In this case the three business applications are essentially copies of each other.

## Tier

Business applications contain tiers. The traffic in a business application flows between tiers. This flow is represented in AppDynamics flow maps along with performance data for the traffic.

An AppDynamics tier represents an instrumented service (such as a web application) or multiple services that perform the exact same functionality, and may even run the same code.

In the AppDynamics model, a tier is composed of one node or multiple redundant nodes. For example the inventory tier may have one node whereas the ecommerce tier may use 3 nodes. Each node in a multi-node tier provides identical functionality, and there is no traffic between nodes.

One example of a multi-node tier is when you have a set of clustered application servers or services.

A difference between a business application and a tier is that there is no interaction between the nodes in a tier. There is interaction between tiers in an application.

Business transactions flow through tiers. This means that there is always a tier that is the starting point for a business transaction. The method call that starts the business transaction is called the entry point.

## Node

A node is the basic unit of processing that AppDynamics monitors. By definition a node is instrumented by an AppDynamics agent, either an app agent or machine agent or both. App agents are installed on JVMs and Windows .NET applications (IIS, executables or services). Machine agents are installed on virtual or physical machine operating systems.

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

## Naming Guidelines

Use names that are recognizable to the people in your group or company.

For details see [Name Business Applications, Tiers, and Nodes](#).

### Learn More

- [Logical Model](#)

## Getting Started

- [Initial Installation](#)
  - [Self-Service Trial or Standard?](#)
  - [On-premise or SaaS?](#)
    - [Get Started with AppDynamics SaaS](#)
    - [Get Started With AppDynamics On-Premise](#)
- [Monitoring, Troubleshooting, and Analyzing Application Performance](#)

This section gives you a roadmap to using AppDynamics.

## Initial Installation

### Self-Service Trial or Standard?

If you are using the self-service trial see [Install Agents for 5 or fewer JVMs or CLR's \(Self-Service Installations\)](#).

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

### On-premise or SaaS?

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

For information about the different approaches see:

- [SaaS and On-Premise Deployment Options](#)
- [SaaS Availability and Security](#)
- [Differences when using a SaaS Controller](#)

### Get Started with AppDynamics SaaS

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

### Get Started With AppDynamics On-Premise

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

## Monitoring, Troubleshooting, and Analyzing Application Performance

To get started using AppDynamics after it is installed see:

- [AppDynamics Essentials](#)

### Get Started with AppDynamics SaaS



*"Deploying APM in the Enterprise... the Path of the Rock Star"*

Follow these steps to get started with AppDynamics.

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

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

## Get Your SaaS Account Information from AppDynamics

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

## Design Your AppDynamics Deployment

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

## Download the AppDynamics Agents

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

- Java Agent
  - .NET Agent
  - Machine Agent
- For details see [Download AppDynamics Software](#).

## Install the AppDynamics Agents

Install agents on the application servers you want to instrument and any other machines you want to monitor. Follow the [instructions to install the AppDynamics Agents](#).

## SaaS Login Credentials

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

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

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

## Connecting Agents to Your SaaS Controller Service

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

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

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

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

## Access the AppDynamics UI from a Browser

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

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

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

## Review the Dashboards and Flow Maps

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

## Review Defaults and Configure Business Transactions, if Needed

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

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

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

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

## Review Default Health Rules and Set Up Policies

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

## Review Default Error Detection

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

## Explore Additional Data and Metric Features

Explore these features to gain more insight into application performance:

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

## Configure Advanced Features

Additional features you may want to use include:

- [End User Monitoring](#)
- [Custom Dashboards](#)
- [Automation](#)
- [System Integrations](#)

## Start Monitoring and Troubleshooting

Start getting the benefits of AppDynamics! See:

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

## Questions?

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

## Use a SaaS Controller

- [Your SaaS Controller URL](#)
- [Login Credentials](#)
- [Connecting Agents to Your SaaS Controller Service](#)
- [SMTP Service for SaaS](#)
- [SaaS Limitations](#)
- [Contact Support](#)

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

### Your SaaS Controller URL

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

The URL is of the following form:

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

### Login Credentials

Login credentials are included in the welcome email from AppDynamics.

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

## Connecting Agents to Your SaaS Controller Service

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

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

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

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

## SMTP Service for SaaS

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

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

The configuration is:

**host:** localhost

**port:** 25

No authentication is needed.

## SaaS Limitations

Compared to an on-premise installation, the following are limitations in the SaaS environment:

- Flow maps show a maximum of the last 60 minutes of data, regardless of whether the Time Range is set to a larger range. Other graphs will display according to the selected Time Range.
- There is a minimum time lag of 4 minutes before data is displayed in the UI.
- Custom action scripts are not supported.
- LDAP integration is limited.
- The business transaction limit is 200 per business application, unless you are using a dedicated Controller.
- Configuration settings related to data retention is limited.

Contact AppDynamics Support for details.

See also <http://www.appdynamics.com/products-saas-on-premise.php>.

## Contact Support

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

## SaaS Availability and Security

- [Service Availability](#)
- [Customer Account Login Security](#)
- [Hosting](#)
- [Data Access](#)
- [Data Collection](#)
- [Data Communication](#)

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

### Service Availability

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

### Customer Account Login Security

The AppDynamics user interface (UI) uses TLS 1.0 with AES 256 bit encryption terminated at the server to ensure end-to-end security over the wire. For additional security, AppDynamics can restrict UI access to customer corporate networks.

### Hosting

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

Every server is operated in a fully redundant fail-over pair to ensure high availability. Data is backed up nightly, stored redundantly and can be restored rapidly in case of failure. AppDynamics also provides an off-site backup service that is available at additional cost.

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

### Data Access

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

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

### Data Collection

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

### Data Communication

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

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

A single agent with the default configuration will typically push between 300KB to 500KB of data per minute depending on application characteristics. AppDynamics uses random staggering on agent data communication to the AppDynamics SaaS platform so traffic is spread evenly to minimize bursts and spikes of network traffic from your data center to the AppDynamics SaaS platform.

# of Agents	Typical Network Bandwidth Used (per min)
1	300KB to 500KB
100	4Mbit to 6.4Mbit
1000	40Mbit to 64Mbit

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

## Get Started With AppDynamics On-Premise





*"Deploying APM in the Enterprise... the Path of the Rock Star"*

Follow these steps to get started with AppDynamics.

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

- Design Your AppDynamics Deployment
- Size and Verify the Controller Environment
- Download AppDynamics
- Install the AppDynamics Controller
- Install the AppDynamics Agents
- Access the AppDynamics UI from a Browser
- Review the Dashboards and Flow Maps
- Review Defaults and Configure Business Transactions, if Needed
- Review Defaults and Configure Databases and Remote Services, if Needed
- Review Default Health Rules and Set Up Policies
- Review Default Error Detection
- Explore Additional Data and Metric Features
- Configure Advanced Features
- Start Monitoring and Troubleshooting

## Design Your AppDynamics Deployment

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

## Size and Verify the Controller Environment

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

## Download AppDynamics

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

## Install the AppDynamics Controller

The AppDynamics Controller is the central management server where all data is stored and analyzed. All AppDynamics Agents connect

to the Controller to report data, and the Controller provides a browser-based user interface for monitoring and troubleshooting application performance. A wizard installs the Controller in just a few minutes. Install the AppDynamics Controller only if you are using the on-premise Controller deployment option.

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

## Install the AppDynamics Agents

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

## Access the AppDynamics UI from a Browser

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

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

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

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

## Review the Dashboards and Flow Maps

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

## Review Defaults and Configure Business Transactions, if Needed

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

- [Business Transaction Monitoring](#)
- [Configure Business Transaction Detection](#)

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

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

## Review Default Health Rules and Set Up Policies

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

## Review Default Error Detection

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

## Explore Additional Data and Metric Features

Explore these features to gain more insight into application performance:

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

## Configure Advanced Features

Additional features you may want to use include:

- [End User Monitoring](#)
- [Custom Dashboards](#)
- [Automation](#)
- [System Integrations](#)

## Start Monitoring and Troubleshooting

Start getting the benefits of AppDynamics! See:

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

## Download AppDynamics Software

- [Accessing the AppDynamics Download Center](#)
  - [Download Tips](#)
- [AppDynamics Software Components](#)
- [Access to Older Versions](#)
- [Downloading from the Linux Shell](#)
- [Learn More](#)

## Accessing the AppDynamics Download Center

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


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

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

## Download Tips

Always copy or transfer the downloaded files in binary mode.

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

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

## AppDynamics Software Components

<b>AppDynamics Software Component</b>	<b>Description</b>	<b>SaaS</b>	<b>On-Premise</b>	<b>Files</b>
Controller	Central management server where all data is stored and analyzed.	N/A	Required	

Java App Server Agent	Instrumentation Agent for Java virtual machines. This component must be installed on each Java application server you want to instrument through AppDynamics.	Required for Java	Required for Java	<ul style="list-style-type: none"> <li>For Sun and JRockit JVMs: AppServerAgent-x.x.x.zip</li> <li>For IBM JVMs: AppServerAgent-ibm-x.x.x.zip</li> </ul>
.NET App Server Agent (includes a Machine Agent by default)	Instrumentation Agent for .NET Common Language Runtime (CLR). This component must be installed on those worker processes that you want to instrument through AppDynamics.	Required for .NET	Required for .NET	<ul style="list-style-type: none"> <li>For Windows 32-bit: dotNetAgentSetup.msi</li> <li>For Windows 64-bit: dotNetAgentSetup64.msi</li> <li>If you are using 32-bit Application on a 64-bit machine, also install the "32-bit add-on for 64-bit" binary.</li> </ul>
Machine Agent	Collects hardware performance metrics and can be installed on any machine in your environment. The Machine Agent can be extended to collect data from other subsystems.	Optional	Optional	MachineAgent-x.x.x.xGA.zip

## Access to Older Versions

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

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

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

## Downloading from the Linux Shell

To download AppDynamics software from a Linux shell, use the wget utility.

Use the following parameters for wget:

```
wget --content-disposition '<URL_TO_FILE>?username=<USERNAME>&password=<PASSWORD>'
```

Ensure that the URL enclosed by single quotes, for example:

```
wget --content-disposition
'http://download.appdynamics.com/onpremise/public/AppDynamics_GA/latest/controller_64bit_linux_v3'
```

The --content-disposition option saves a file with a proper name that does not have the query string attached to the file name. On some Linux environments, wget may not have the content-disposition option. In this case, use following parameters for wget:

```
wget
'[http://download.appdynamics.com/onpremise/public/AppDynamics_GA/latest/controller_64bit_linux_v3']
```

## Learn More

- [Supported Environments and Versions](#)
- [Agent - Controller Compatibility Matrix](#)

## Quick Start for DevOps

## Get Started

[Get Started on SaaS](#)

[Get Started On-Premise](#)

[Features Overview](#)

## Tutorials for Java

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

[Understanding Events](#)

[Understanding Flow Maps](#)

[Understanding Slow Transactions](#)

[Understanding the Transaction Scorecard](#)

[Understanding Server Health](#)

[Understanding Exceptions](#)

## Learn More

[Best Practices for Application Developers](#)

[Best Practices for Performance and Quality Assurance Engineers](#)

[Best Practices for Operations Professionals](#)

## Monitor Your Applications

[Business Transaction Monitoring](#)

[Monitor End User Experience \(EUM\)](#)

[Monitor Events](#)

[Monitor Application Change Events](#)

[Background Task Monitoring](#)

[Backend Monitoring](#)

[Infrastructure Monitoring](#)

[Monitor CLR's](#)

[Monitor Hardware](#)

## Troubleshoot Application Performance

[Troubleshoot Slow Response Times](#)

[Troubleshoot Health Rule Violations](#)

[Transaction Snapshots](#)

[Troubleshoot Node Problems](#)

[Troubleshoot Errors](#)

[Diagnostic Sessions](#)

[Troubleshoot Java Memory Issues](#)

## Quick Start for Architects

### Get Started

[Supported Environments and Versions](#)

[Get Started on SaaS](#)

[Get Started On-Premise](#)

### Concepts

[Features Overview](#)

[Architecture](#)

[Logical Model](#)

[Mapping Application Services to the AppDynamics Model](#)

Behavior Learning and Anomaly Detection  
Thresholds  
Glossary

## Basic Configuration

Configure Business Transaction Detection  
Configure Policies  
Configure Baselines  
Configure Thresholds  
Configure Error Detection  
Configure End User Experience  
Alerting Wizard  
Custom Dashboards

## Analyze

Business Metrics  
Infrastructure Metrics  
Reports  
Compare Releases

## Learn More

## Advanced Configuration

Hierarchical Configuration Model  
Configure Data Collectors  
Configure Code Metric Information Points  
Configure Backend Detection  
Configure Call Graphs  
Configure Background Tasks  
Configure Stale Backend Removal  
Configure Custom Memory Structures (Java)  
Configure JMX Metrics from MBeans  
Configure Multi-Threaded Transactions (Java)  
Configure Object Instance Tracking (Java)  
Configure Transaction Snapshots  
Configure Memory Monitoring (Java)  
Internationalization  
Integrate using Custom Action Scripts  
Export and Import Business Application Configurations

## Integration

System Integrations  
Use the AppDynamics REST API

## Automation

Workflow Automation  
Workflow for Cloud Orchestration

## Quick Start for Administrators

## Get Started

Basic Components  
Get Started on SaaS  
Get Started On-Premise  
Basic Concepts

## Basic Administration

Release Notes for AppDynamics Pro  
Supported Environments and Versions  
Install and Upgrade AppDynamics  
Name Business Applications, Tiers, and Nodes

## Learn More

### Advanced Administration

Implement SSL  
Best Practices for Failover Scenarios  
Administer Agents  
Administer the Controller  
Configure Authentication, User Permissions and Integrations  
AppDynamics for Large Enterprises

## Quick Start for Operators

### Get Started

AppDynamics Essentials

### Tutorials for Java

Use AppDynamics for the First Time with Java  
Understanding Events  
Understanding Flow Maps  
Understanding Slow Transactions  
Understanding the Transaction Scorecard  
Understanding Server Health  
Understanding Exceptions

## Learn More

Best Practices for Operations Professionals  
Set User Preferences  
Custom Dashboards

## Set User Preferences

- Accessing My Preferences
  - To user preferences
  - Account Information
  - Advanced Features
  - View Preferences
    - To set Help Popups
    - To change the navigation panel color scheme
    - To change the Metric Browser graph color scheme
    - To change the Dashboard graph color scheme
    - To specify the minimal number of backends to display
    - To set the date format
    - To use the 24 hour time format
    - To enable debug mode
    - To display server call data
    - To enable server call logging

This topic describes how users can change their password and set various preferences in the My Settings view.

## Accessing My Preferences

### To user preferences

1. Click the Setup menu in the upper right section of the screen:
2. From the drop-down menu, click **My Preferences**.

The My Preferences screen opens.

### Account Information

In the My Account pane, you can update your user name and password.

### Advanced Features

If you want to access AppDynamics cloud automation features, such as workflows, check the Show Cloud Auto-Scaling features checkbox. The Automate menu appears in the left navigation pane.

Automation features are triggered by policy actions.

See [Workflow Automation](#) for information about automation. See [Policies](#) for information about policies.

### View Preferences

The View Preferences pane lets you configure various preferences in the user interface.

#### To set Help Popups

By default Help popups are enabled. Not every screen has these but those that do will display the tips every time you open the screen. When that happens, on a per-screen basis you can check Don't Show Again. If you've checked Don't Show Again for many screens and want to re-enable them, check Enable Help Pop-ups and click **Reset All**.

To disable Help popups for the whole UI at all times, clear **Enable Help Pop-ups** on the View Preferences pane.

#### To change the navigation panel color scheme

Select either Light or Dark from the Navigation Panel Color Scheme pulldown.

#### To change the Metric Browser graph color scheme

Select either Light or Dark from the Metric Browser Graph Color Scheme pulldown.

#### To change the Dashboard graph color scheme

Select either Dark or Light from the Dashboard Graph Color Scheme pulldown.

#### To specify the minimal number of backends to display

Enter a number in the Maximum number of backends field. The default is 20.

#### To set the date format

Select a format from the **Date Format** pulldown menu.

#### To use the 24 hour time format

Click the **Use 24 Hour Format** check box.

#### To enable debug mode

Click the **Enable Debug Mode** check box.




## To display server call data

Click the **Display Server call data in the lower right of the screen** check box.

## To enable server call logging

Click the **Enable Advanced Server Call Logging** check box.

 Warning: Enabling call logging can slow down the CPU, especially if there are large datasets.

# Glossary

- Action
- Agent
- Alert
- Alert Digest
- Anomaly Detection
- Ajax Request
- Application Performance Management
- APM
- App Server
- App Server Agent
- Application
- Application Dashboard
- Backend
- Background Task
- Baseline
- Baseline Deviation
- Baseline Pattern
- BCI
- Browser Snapshot
- Business Application
- Business Transaction
- Bytecode Injection
- Call Graph
- Compute Cloud
- Controller
- Detection
- Diagnostic Session
- Discovery
- Distributed Application
- Entry Point
- Error
- Error Transaction
- Exception
- Event
- Exit Point
- Flow Map
- Health
- Health Rule
- Health Rule Violation
- High Availability (HA) Cluster
- Histogram
- iFrame
- Information Point
- Key Performance Indicator
- KPI
- Machine
- Machine Agent
- Managed Application
- Match Condition
- Node
- Pageview
- Policy
- Remote Service
- Request
- Scorecard

- Task
- Tier
- Tag and follow
- Threshold
- Trace, tracing
- Transaction Snapshot
- Workflow

### **Action**

An action automatically responds to an event, usually without operator interaction. There are various types of actions which can be notification, diagnostic, or remedial.

### **Agent**

In AppDynamics an agent collects data about an application (and optionally machine) performance. AppDynamics has application server agents (app agents) and machine agents. A Java agent intercepts the bytecode loading at the classloader and enhances it before it is loaded in the JVM. See also Bytecode Injection.

### **Alert**

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

### **Alert Digest**

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

### **Anomaly Detection**

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

### **Ajax Request**

An Ajax request is a request for data sent from a page using Asynchronous JavaScript and HTML (Ajax). The Ajax request is tracked as a child page using EUM.

### **Application Performance Management**

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

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

### **APM**

See Application Performance Management.

### **App Server**

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

### **App Server Agent**

An app server agent monitors an application server. A Java app server agent monitors a JVM. A .NET app server agent monitors a CLR. See also Node.

### **Application**

See Business Application.

## ***Application Dashboard***

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

## ***Backend***

A backend or backend node is not instrumented directly, but you can monitor the flow of traffic to it. Typical examples are a database or messaging service.

## ***Background Task***

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

## ***Baseline***

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

## ***Baseline Deviation***

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

## ***Baseline Pattern***

A baseline pattern defines how a dynamic baseline is calculated. Time period baselines use a fixed range of time ("1/1/2011 - 1/31/2011"). Trend baselines use a rolling time range (last 3 months, last year).

## ***BCI***

See Bytecode Injection.

## ***Browser Snapshot***

A browser snapshot presents set of diagnostic data for an individual base page, iFrame or Ajax request. The data reports the end-user's experience starting with the Web browser. Browser snapshots are taken at a certain point in time.

## ***Business Application***

A business application serves some major business functionality. In AppDynamics a business application does not necessarily map one-to-one to a single Java/J2EE application (WAR / EAR) or to a single .NET application (IIS/ASP.NET).

## ***Business Transaction***

A business transaction represents an aggregation of similar user requests to accomplish a logical user activity. Examples of these activities include: logging in, searching for items, adding items to the cart, checking out (e-commerce); content sections that users navigate such as sports, world news, entertainment (content portal); viewing a quote, buying and selling stocks, placing a watch (brokerage). A single request is a business transaction instance.

## ***Bytecode Injection***

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

## ***Call Graph***

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

## ***Compute Cloud***

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

## ***Controller***

The Controller collects stores, baselines and analyzes performance data collected by agents. Can be installed on-premise or using the AppDynamics SaaS model.

## ***Detection***

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

## ***Diagnostic Session***

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

## ***Discovery***

See Detection.

## ***Distributed Application***

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

## ***Entry Point***

An entry point begins a business transaction. Typically an entry point is a method or operation. Auto-detected for most common frameworks but configurable for customized business transaction detection.

## ***Error***

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

## ***Error Transaction***

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

## ***Exception***

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

## ***Event***

An event represents a change in application state. There are different event types.

## ***Exit Point***

An exit point initiates an external call from an app server to a database, remote service or to another app server. An exit point is auto-detected at startup and is configurable for customized backend detection.

## ***Flow Map***

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

## ***Health***

A visual summary of the extent to which a business transaction or server is experiencing critical and warning health rule violations.

## ***Health Rule***

A health rule defines a condition or set of conditions in terms of metrics. The condition compares the performance metrics that

AppDynamics collects with some static or dynamic threshold that you define. If performance exceeds the threshold, a health rule violation event is triggered.

### ***Health Rule Violation***

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

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

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

### ***Histogram***

A histogram graphically represents a frequency distribution using rectangles whose widths represent class intervals and whose areas are proportional to the corresponding frequencies.

### ***iFrame***

An iFrame is an HTML element embedded in another HTML element and is tracked as a child page using EUM.

### ***Information Point***

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

### ***Key Performance Indicator***

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

### ***KPI***

See Key Performance Indicator.

### ***Machine***

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

### ***Machine Agent***

A machine agent instruments a machine to report data about hardware and the network to the Controller.

### ***Managed Application***

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

### ***Match Condition***

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

### ***Node***

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

### ***Pageview***

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

### ***Policy***

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

### ***Remote Service***

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

### ***Request***

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

### ***Scorecard***

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

### ***Task***

A task codifies a unit of work as a set of instructions. Component of a step in a workflow

### ***Tier***

A tier represents a key service in an application environment, such as a website or processing application. A tier is composed of one or more nodes. See Node.

### ***Tag and follow***

Tag and follow is a methodology used for tracing code execution.

### ***Threshold***

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

### ***Trace, tracing***

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

### ***Transaction Snapshot***

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

### ***Workflow***

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

## **AppDynamics Support**

If you have questions about using AppDynamics please [file a support ticket \(http://help.appdynamics.com/tickets/new\)](http://help.appdynamics.com/tickets/new) and attach any logs that are related to your issue.

To get log files from the Controller, see [Log Files for Troubleshooting](#).

To get heap, histogram, and thread dumps see [Controller Dump Files](#).

If you want the Support team to remotely monitor your Controller, see [Configure Remote Monitoring of an On-Premise Controller](#).

## Controller Dump Files

- Controller Troubleshooting Information Needed by the AppDynamics Support Team
  - To get the heap and histogram dump files
  - To take four thread dumps at three second intervals
  - Provide the AppDynamics Support Team with the files

## Controller Troubleshooting Information Needed by the AppDynamics Support Team

If requested, collect the following files to send to the AppDynamics Support Team.

### To get the heap and histogram dump files

- Get the **process id of the Controller** to use in subsequent commands.

```
ps -ef | grep java
```

- Get the **heap dump before garbage collection** using following command:

```
<java-jdk-install-dir>/bin/jmap -dump:format=b,file=heap_before_live.bin  
<Controller_pid>
```

- Get the **histogram before garbage collection** using following command:

```
<java-jdk-install-dir>/bin/jmap -histo <Controller_pid> | head -200 >  
histo_before_live.txt
```

- Get the **histogram after garbage collection** using following command:

```
<java-jdk-install-dir>/bin/jmap -histo:live <Controller_pid> | head -200 >  
histo_after_live.txt
```

2. Save the generated files:

- heap\_before\_live.bin
- histo\_before\_live.txt
- histo\_after\_live.txt

### To take four thread dumps at three second intervals

- Using the Controller process ID, execute following command:

```
kill -3 <Controller_pid>
```

- Save the <Controller\_Installation\_Directory>/appserver/domains/domain1/logs/jvm.log file.

### Provide the AppDynamics Support Team with the files

- heap\_before\_live.bin
- histo\_before\_live.txt
- histo\_after\_live.txt
- jvm.log

## Log Files for Troubleshooting

- [Log Files that Record Possible Problems](#)

Use installer.log and the files in the Controller logs directory to troubleshoot Controller errors.

### Log Files that Record Possible Problems

The following log files may reveal errors:

- **installer.log:**  
This file contains information about events of the install process such as extraction, preparation and other post-processing tasks. It is located at <Controller\_Installation\_directory>.
- **server.log:**  
This file contains log information for the embedded Glassfish application server used by the Controller. It is located at <Controller\_Installation\_directory>/logs.
- **database.log:**  
This file contains log information for the MySQL database that is used by the Controller. It is located at <Controller\_Installation\_directory>/logs.
- **installation.log**  
This file contains log information about the installation. It is located at <Controller\_Installation\_directory>/install4j.

If after analyzing the logs you have questions, please [file a support ticket](#) and attach any logs that are related to your issue. See [AppDynamics Support](#) for suggestions.

## Configure Remote Monitoring of an On-Premise Controller

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

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

### Prerequisites

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

### Configuring Remote Monitoring

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

#### Make a formal request to AppDynamics Support

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

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

- Account name
- Account key
- Application name



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



## Shut down your Controller and its application server

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

- For Linux:

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

- For Windows:

```
controller.bat stop
```

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

For Linux:

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

For Windows:

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

## Modify the Controller configuration

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

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

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

If SSL is required add:

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

See [Controller SSL and Certificates](#).

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

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


1. Create a new controller-info.xml file. Refer to the sample controller-info.xml file shown below.  
If SSL is required set controller.ssl.enabled to "true".

2. Add the new controller-info.xml file to the <Controller\_Installation\_Directory>/appserver/domains/domain1/appagent/conf folder.

3. Set the account name to the account name supplied by AppDynamics Support.

4. Set the application name to the application name supplied by AppDynamics Support.

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

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

### Sample controller-info.xml file

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

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

### Restart the Controller's application server

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

- For Linux:

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

- For Windows:

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

### Install the Machine Agent for the Controller

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

### Monitor the performance of the Controller

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

## Learn More

- [Controller SSL and Certificates](#)

## Download Doc PDFs

### Release Notes

[Release Notes \(337 kB\)](#) Updated for 3.7 on May 13, 2013

### All Docs

[Complete Documentation Suite \( MB\)](#)

### Doc Subsets

[Release Notes \(337 kB\)](#) Updated May 13, 2013

[AppDynamics Essentials \(904 kB\)](#)

[AppDynamics Features \(13,159 kB\)](#)

[Get Started \(121 kB\)](#)

[Get Started with AppDynamics SaaS \(84 kB\)](#)

[Get Started With AppDynamics On-Premise \(74 kB\)](#)

[Introduction and Tutorials \(5.87 MB\)](#)

[Use AppDynamics \(13.31 MB\)](#)

[AppDynamics for Java \(14,664 kB\)](#)

[AppDynamics for .Net \( 1,161 kB\)](#)

[End User Experience \(2.82 MB\)](#)

[Configure AppDynamics \(7,379 kB\)](#)

[Integrate With Other Systems - includes REST and 3rd Parties \(699 kB\)](#)

[Monitor \(7,040 kB\)](#)

[Alert and Respond \(1.22 MB\)](#)

[Alert and Automate \(5.49 MB\)](#)

[AppDynamics Administration \(3.11 MB\)](#)

[Automate AppDynamics \(354 kB\)](#)

[Troubleshoot \(1,034 kB\)](#)

[AppDynamics Best Practices \(3,359 kB\)](#)

## Use the Documentation Wiki

- [Locate Information Using Search](#)
- [Locate Information Using Labels](#)
- [Use Comments for Feedback](#)
- [Generate a PDF or Word File from a Page](#)
- [Download a PDF of the Documentation](#)
- [Generate PDF or HTML Versions of Pages](#)

- [Known Issues](#)
  - "Duplicate headers received from server" Error on PDF Export

## Locate Information Using Search

1. Type a word or phrase in the Search box located at the upper right of the wiki pages.
2. Scroll through the list of results.

## Locate Information Using Labels

Labels are keywords or tags that are added to pages. To view the labels defined in the system,

1. Click **Browse -> Labels**. The wiki displays a list of popular labels.
2. Click on any label to find pages tagged with that label.

Note: Labels are handy but may not return all instances of a topic. Use Search for more fine-grained results.

## Use Comments for Feedback

AppDynamics is happy to receive your feedback! We strongly encourage you to leave comments on pages. Scroll to the bottom of a page and click **Add Comment** to add your comment. Comments are regularly monitored.

## Generate a PDF or Word File from a Page

1. Navigate to the page.
2. Click **Tools -> Export to PDF** or **Tools -> Export to Word**.

## Download a PDF of the Documentation

The entire documentation is available to download in PDF format as a ZIP file.

1. In the left navigation bar, scroll to the bottom.
2. Click the most recent ZIP file.

## Generate PDF or HTML Versions of Pages

You can generate PDF or HTML versions of part or all of the documentation.

1. Go to the "Browse" menu.
2. Click **Browse -> Advanced**.
3. Click **PDF/HTML Export**.
4. Select those pages that you want to be available offline.
5. Click on "Export" button.



## Known Issues

### "Duplicate headers received from server" Error on PDF Export

When exporting a single page as PDF the follow error occurs:

Duplicate headers received from server  
The response from the server contained duplicate headers. This problem is generally the result of a misconfigured website or proxy. Only the website or proxy administrator can fix this issue.  
Error 349 (net::ERR\_RESPONSE\_HEADERS\_MULTIPLE\_CONTENT\_DISPOSITION): Multiple distinct Content-Disposition headers received. This is disallowed to protect against HTTP response splitting attacks.

This is most likely due to a problem in the Chrome browser. In Chrome, files with non-standard alpha-numeric characters in the file name, such as , \$ % & \* characters, can cause this issue. Try another browser.

## License Information

In the Controller UI, click **Setup -> License** to see information about the AppDynamics licenses installed on your system.

The License window shows the name of your customer account and the type of license (Trial, etc.) used for the account. It lists the

number of agent licenses and expiration date.

The App Agent for Java consumes a license when the JVM is instrumented, regardless of whether the node is used in the UI. To remove an agent license see [Manage App Agents#Deleting App Agents](#).

- [Controller Licenses](#)
- [EUM License](#)

## Documentation Map

### Get Started

[Get Started on SaaS](#)

[Get Started On-Premise](#)

[Features Overview](#)

[Architecture](#)

[Logical Model](#)

[Quick Start for DevOps](#)

[Quick Start for Operators](#)

[Quick Start for Administrators](#)

[Quick Start for Architects](#)

### Tutorials for Java

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

[Understanding Events](#)

[Understanding Flow Maps](#)

[Understanding Slow Transactions](#)

[Understanding the Transaction Scorecard](#)

[Understanding Server Health](#)

[Understanding Exceptions](#)

### Monitor and Troubleshoot

[Best Practices for Application Developers](#)

### Monitor Your Applications

[Business Transaction Monitoring](#)

[Monitor End User Experience \(EUM\)](#)

[Monitor Events](#)

[Monitor Application Change Events](#)

[Background Task Monitoring](#)

[Health Rules](#)

### Troubleshoot Application Performance

[Troubleshoot Slow Response Times](#)

[Troubleshoot Health Rule Violations](#)

[Transaction Snapshots](#)

[Troubleshoot Node Problems](#)

[Troubleshoot Errors](#)

[Diagnostic Sessions](#)

[Troubleshoot Java Memory Issues](#)

### Analysis

Business Metrics  
Infrastructure Metrics  
Reports  
Compare Releases

## Administration

### Basic Administration

Release Notes for AppDynamics Pro  
Supported Environments and Versions  
Install and Upgrade AppDynamics  
Name Business Applications, Tiers, and Nodes

### Advanced Administration

AppDynamics for Large Enterprises  
Implement SSL  
Best Practices for Failover Scenarios  
Administer Agents  
Administer the Controller  
Configure Authentication, User Permissions and Integrations

## Concepts

Architecture  
Logical Model  
Mapping Application Services to the AppDynamics Model  
Behavior Learning and Anomaly Detection  
AppDynamics for Large Enterprises  
Glossary

## Configuration

### Basic Configuration

Configure Business Transaction Detection  
Configure Policies  
Configure Baselines  
Configure Thresholds  
Configure Error Detection  
Configure End User Experience  
Alerting Wizard  
Custom Dashboards

### Advanced Configuration

Hierarchical Configuration Model  
Configure Data Collectors  
Configure Code Metric Information Points  
Configure Backend Detection  
Configure Call Graphs  
Configure Background Tasks  
Configure Stale Backend Removal  
Configure Custom Memory Structures (Java)  
Configure JMX Metrics from MBeans  
Configure Multi-Threaded Transactions (Java)  
Configure Object Instance Tracking (Java)  
Configure Transaction Snapshots  
Configure Memory Monitoring (Java)  
Internationalization

Integrate using Custom Action Scripts  
Export and Import Business Application Configurations

## Integration

System Integrations  
Use the AppDynamics REST API

## Automation

Workflow Automation  
Workflow for Cloud Orchestration

## Reference

### User Interface Reference

Dashboards  
Transaction Analysis Tab  
Flow Maps  
Time Ranges  
Call Graphs  
Scorecards  
KPI Graphs  
Key to the AppDynamics Icons

## FAQs

Controller Install and Admin FAQ  
Controller High Availability FAQ  
Machine Agent Install and Admin FAQ

## AppDynamics Support

Controller Dump Files  
Log Files for Troubleshooting  
Configure Remote Monitoring of an On-Premise Controller  
Download Doc PDFs  
Use the Documentation Wiki